AFRL-IF-RS-TR-1998-147
Final Technical Report
July 1998

MLS DIRECTORY SERVER PROTOTYPE

Infosystems Technology, Inc.

Charles J. Testa

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

AFRL-IF-RS-TR-1998-147 has been reviewed and is approved for publication.

APPROVED: Mary L. Denz

MARY L. DENZ
Project Engineer

FOR THE DIRECTOR: [illegible]

WARREN H. DEBANY, Jr., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE Jul 98	3. REPORT TYPE AND DATES COVERED Sep 95 - Sep 97

4. TITLE AND SUBTITLE MLS DIRECTORY SERVER PROTOTYPE 6. AUTHOR(S) Charles J. Testa	5. FUNDING NUMBERS C - F30602-95-C-0298 PE - 33140F PR - 7820 TA - 04 WU - 30
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Infosystems Technology, Inc. 6411 Ivy Lane Greenbelt, MD 20770	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/IFGB 525 Brooks Rd. Rome, NY 13441-4505	10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-1998-147

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Mary L. Denz/IFGB/315-330-2030

12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This is the final report for the congressional funded Multilevel Secure (MLS) Directory Server Prototype program. Infosystems Technology, Inc., Wang Federal, Inc., J.G. Van Dyke & Associates, Inc., and Datacraft Limited have been working on the design and development of a proof-of-concept MLS Directory Server for the Air Force Research Laboratory, Rome Research Site. This document summarizes the work performed and design features of this effort, and some of the technical issues that prevented the successful integration of a working MLS Directory Server within the budgetary constraints of this contract. Despite non-completion, the technical knowledge gained from this prototype will be valuable when it comes time to develop a fully functional MLS Directory server based on the X.500 International Standards.

14. SUBJECT TERMS Computer Security, Trusted Database Management System, X.500, Trusted Computer Base, Multilevel Secure Directory System	15. NUMBER OF PAGES 124 16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

TABLE OF CONTENTS

1 Introduction

Infosystems Technology, Inc. (ITI), Wang Federal, Inc, J.G. Van Dyke & Associates, Inc. and Datacraft Limited have been working on the design and development of a proof-of-concept Multi-level Secure (MLS) Directory Server for the U.S. Air Force's Rome Laboratories. This document summarizes the work performed and design features of this effort, and some of the technical issues that prevented the successful integration of a working MLS Directory Server within the budgetary constraints of this contract.

Despite non-completion, the technical knowledge gained from this prototype will be valuable when it comes time to develop a fully functional MLS Directory System based on the X.500 International Standards, e.g., for the NSA MISSI program. The need for X.500 directories within DoD has grown tremendously with the implementation of the Defense Message System and other large networking initiatives. X.500 provides a global repository for storage of virtually any type of information. As information is gathered and maintained in support of the DoD it is rapidly becoming apparent that this information needs to be protected.

This proof-of-concept was intended to be the first step in providing protection of X.500 information. It was to use the XTS-300™ B3-evaluated Trusted Computer Base developed by Wang Government Services as its platform system, the not-yet-evaluated Trusted RUBIX™ relational database management system developed by Infosystems Technology, Inc., the J.G. Van Dyke & Associates, Inc. FORTEZZA® strong authentication software, and the Datacraft Limited X.500 Directory System Agent (DSA) to provide the highest available security mechanisms.

The objective of the proof-of-concept was to provide a MLS Directory Server that would support standard X.500-1993 DSA functionality and protocols, and which would appear to standard X.500 Directory User Agents, Administrative Directory User Agents, and Directory System Agents like any other standard DSA. The difference was that the MLS Directory Server would be able to store information at several classification levels and/or multiple compartments/categories. In this way, the MLS Directory Server would have maintained directory information at its true classification level, enabling it to eliminate the multiplicity of system-high directories which often contain large amounts of replicated lower-level information that is necessarily overclassified in system-high mode.

The security policy of the MLS Directory Server — enforced by the Trusted Computing Bases (TCBs) of the B3 evaluated XTS-300 and the Trusted RUBIX RDBMS (designed to be "B3 capable") — would have ensured that each external X.500 system could access only those levels in the MLS Directory information base dominated by the requestor's own authorization level. In this way, a user's directory entry would have been able to contain data at several classification levels, yet be stored and administered on a single Directory Server. The MLS Directory Server would have restricted lookups and updates of that directory entry to only those portions of the entries which the requestor was allowed to access.

2 Work Performed

The following sections describe the work performed in this project, including tasks successfully completed, those that were undertaken but not completed, and those originally planned but not undertaken.

2.1 Tasks Completed

The following tasks were successfully completed:

1) Defined and delivered requirements specification (see Appendix A), design (see Appendix B), and management plan for the proof-of-concept.

2) Performed market survey of X.500 products to determine which DSA would best meet the requirements of this project.

3) Negotiated pricing and support for DSA software, including source code licence.

4) Defined and delivered initial Directory Schema to be supported (see Appendix C).

5) Defined initial Test Plan and Demonstration Scenarios (see Appendix D).

6) Tested DSA communications capabilities with other X.500 DSA and DUA products (from Nexor, ISOCOR, CDC) to ensure interoperability and compliance to the X.500 standards.

7) Ported X.500 DSA, using remote database pass through only, to XTS-300 running STOP Version 4.3.

8) On Sun SPARC system, tested full DSA functionality, including relational database interface, using the Ingres RDBMS.

9) On Sun SPARC system, tested full DSA functionality, including relational database interface, using the Trusted RUBIX RDBMS.

10) Ported Trusted RUBIX to XTS-300, running STOP 4.4.

2.2 Tasks Undertaken But Not Completed

1) Integrated VDA FORTEZZA® strong authentication software with Datacraft X.500 DUA and DSA. Not completed due to shortcomings in version of Datacraft code available to us on this project. Unable, due to budgetary constraints, to upgrade to new Datacraft release, wherein Datacraft took Van Dyke's code and integrated it with their standard DSA product; this upgrade would have enabled demonstration of strong authentication interface.

2) Porting of Datacraft DSA, using local database capability, to XTS-300 running STOP 4.4. Apparent memory allocation errors made it impossible to complete testing of the DSA's relational database interface modules. Due to lack of time and personnel and financial resources, the DSA port to STOP 4.4 was abandoned.

2.3 Tasks Not Undertaken

1) Creation of MLS test directory information base.

2.4 Details Pertaining to Work Performed

2.4.1 Strong Authentication

1) Van Dyke added a command to the definition of *remote-dsa* in the Datacraft init files for *dsa-min-auth*. This allows the DSA to do different types of authentication when binding to different DSAs it knows about.

2) Van Dyke added the Distinguished Name (DN) and time to the bind-token.

3) Van Dyke added the ability for the DSA get its own name for strong authentication from the FORTEZZA card when the DSA logs in.

4) Van Dyke added printing out of the DSA's DN to be used with strong authentication when doing an *assoc_get_config*.

5) Van Dyke finished *assoc_get_users* to know about strong authenticated binds.

6) Van Dyke expanded the size of the session protocol data unit (SPDU) that the DSA would accept upon a session connect.

7) Van Dyke began testing DSP binds.

8) Van Dyke performed initial integration of their developed FORTEZZA strong authentication software with the Datacraft DSA on the XTS-300.

Though not completed for this proof-of-concept, these efforts were not wasted, because the Van Dyke strong authentication code has been incorporated by Datacraft into their standard DSA product, which is being used by the MISSI DMS/DII Guard.

2.4.2 X.500 Product Selection

The DSA to be used on this proof-of-concept had to meet the following requirements:

1) Must support the 1993 Version of the X.500 Series of Recommendations (including access controls, extended information model, replication using shadowing, schema enhancements);

2) DSA must provide SQL interface for storage of directory information;

3) DSA must support ACP 133, or vendor must commit to enhance DSA to provide this support (including integrity on all requests, results, and errors; confidentiality of stored attributes; rule based access control);

4) Vendor must be willing to migrate to Version 3 certificates;

5) Vendor must be willing to provide source code to be ported to the Wang XTS-300 platform for this effort; source code licence will be reduced significantly in price or waived;

6) Vendor must make commitment to support development of the MLS Directory Server;

7) Vendor must be willing to provide source code for review by official government or government contracted certification and accreditation/evaluation facilities such as the U.S. National Computer Security Center.

The Datacraft Directory System Agent (DX500) was selected for this proof-of-concept because it met or exceeded the above requirements. Other DSA products evaluated were the ISODE QUIPU DSA, the Nexor DSA, the Marben DSA, the Unisys DSA. The first three were rejected due to lack of relational database interface or acceptable schedule for providing such an interface; the last was rejected due to lack of support for full DAP (vs. LDAP) and Unisys unwillingness to significantly reduce their source license fee.

Despite the non-completion of the proof-of-concept, the selection of Datacraft's DSA turned out to be a boon. The DMS Guard project began initially using the Nexor DSA, but the highly favorable experiences of Wang's and Van Dyke's engineers with the very well engineered Datacraft software, and the DSA's capabilities, and Datacraft's willingness and commitment to provide ACP-133 and other needed enhancements to their standard commercial product led the DMS Guard project to switch to the Datacraft DSA early in their project.

2.4.3 Port of Trusted RUBIX to XTS-300

The following describes changes made to Trusted RUBIX, difficulties encountered, and optimizations performed for the XTS-300 porting effort.

1) The STOP operating system is not UNIX. All System V "UNIXisms" in the Trusted RUBIX code had to be removed. Trusted RUBIX was designed to take advantage of the underlying System V operating system by directly calling many functions taken for granted by UNIX programmers. To be a truly portable system, Trusted RUBIX had to have these characteristics isolated. These characteristics include, but are not limited to the following:

    - lid_t and level_t data types
    - lvldom, lvlequal, lvlfile, lvlin, lvlout, lvlproc, lvlvfs
    - fork/pipe/exec
    - strdup, getuid, getpid, etc.

2) ITI's technical director made the decision to have the Trusted RUBIX MAC Server act as an auditing tool. Otherwise, the Trusted RUBIX SQL engine component, the largest and most complex part of Trusted RUBIX, would have had to be implemented in STOP Ring 2 (Trusted Software) so it could perform auditing. Keeping the SQL engine in Ring 3 (Untrusted Applications) was very beneficial, both from a security and a portability standpoint. The former because it minimized the Trusted RUBIX trusted computing base (TCB), and the latter because Ring 3 runs on top of the STOP Commodity Application System Services which provide a fairly complete set of Unix System V Application Programmatic Interfaces, instead of the proprietary Trusted Systems Services APIs that run under Trusted Software in Ring 2. As a result, Trusted RUBIX on the XTS-300 was implemented with a call from the SQL engine to the MAC Server to implement auditing.

3) STOP 4.3 provided no mechanism for an untrusted (Ring 3) process to invoke a trusted (Ring 2) process. Wang's engineer developed a "hack" to provide this MLS interprocess communications mechanism (the Trusted Peer Daemon). However, the TPD was determined to be insufficient. The kind of MLS IPC mechanism required, however, was being implemented for the DMS Guard program as part of the Kernel in the next release of the STOP operating system, so it was decided to wait until that release became available before proceeding with the Trusted RUBIX port.

4) The compiler that comes with the STOP operating system has significant limitations. For instance, it took quite a bit of effort to discover that one cannot compile and generate object code with debugging information (-g). Also, to use memory manipulation functions (memcpy, memcmp, etc.) the developer must include <memory.h> to avoid strange compilation errors that do not indicate memory.h is required.

5) Initially, there was no generic standard IPC mechanism between all separate Trusted RUBIX processes. Each process had its own custom version of IPC. To enhance portability, a generic interface for IPC within Trusted RUBIX was built, implemented first under UNIX, then later under STOP. This IPC package was required to complete the port. It was determined that under STOP, a Wang engineer could implement the mechanism more efficiently. The ITI engineer provided specification (man pages), to which specifications the Wang engineer developed the IPC. However, due to lack of testing between processes running in different rings of STOP, the usefulness of the resulting IPC was severely limited.

6) The Trusted RUBIX object isolation mechanism had to be modified because STOP does not allow a UNIX-like "change process level" privilege. There was a significant enough difference between the STOP B3-evaluated protection scheme and the System V protection scheme to require the change.

7) The XTS-300 i486 EISA-based platform, where almost all of the development occurred, was not terribly performant. The lack of performance resulted in slow debug, compile, and run times. ITI later learned that Wang's STOP system developers use a cross-compilation environment for their development efforts, thus avoiding such performance problems.

8) The largest impediment to the progress of the Trusted RUBIX port was the lack of runtime debugging utilities on the XTS-300. A porting problem that would have taken 30-60 minutes to debug took over a week of concerted effort to troubleshoot. This meant the Trusted RUBIX porting effort was drawn out significantly, with problems that would have necessitated one normal work day of debugging on another system taking up to eight weeks to resolve on the XTS-300. printf(), the primary available means for debugging in STOP does not work correctly under Ring 2.

9) The Unix System 3.2 Bourne shell on the XTS-300 crashed repeatedly. XTS-300 developers have generated a gnu based package which provides a modern useable shell (bash), but for internal use only.

10) The STOP O/S libraries are not developer-oriented. Because there is a limited number of engineers using STOP as a development platform, and the STOP developers themselves use a cross-compilation environment, there has not been a large base of libraries developed over the years. For example, there are no library functions to:

- convert MAC labels to strings and visa versa;
- provide communications between rings.

11) The ITI engineers replaced the Trusted RUBIX pipe IPC format with a shared memory IPC.

12) The ITI engineers identified other areas which would make future Trusted RUBIX ports easier. These included adjustments to the organization of Trusted RUBIX to place the MAC TCB components into another set of directories. This directory would correspond to Ring 2 executables (which need to be compiled differently). This work of separating the executables was identified but not performed under this project.

13) The biggest problem, e.g., the show-stopper, was the fact that the DSA port to the XTS-300 did not work. The available version of the DSA was functional as far as the "passthrough" mode was concerned (pass-through mode is when the DSA routes requests to a remote database instead of a local one). It is important to note that the local database mode exercises some portions of DSA code not exercised otherwise, including the DIP (database interface), DIT (database translation), some memory allocation/listing modules, etc. Although the Trusted RUBIX software works when the same SQL queries are typed in manually, running it through the DIP interface causes problems. The problems manifest themselves as errors in the memory allocation modules. Lack of funding prevented debugging of the system by someone knowledgeable of the DSA source code. As noted previously, "blind-debugging" would be difficult and very time-intensive.

14) Some of the functionality of the Datacraft DSA is generated via a program known as an ASN.1 compiler. This compiler takes a descriptive file as input and generates C code as output. On the XTS-300 platform, there is no ASN.1 compiler available. Therefore, all ASN.1 compilations occurred on the Sun SPARCstation, which were then copied over to the XTS-300 for C code compilation and finally linked into the DSA program. This means that there is no method available to natively build the DSA on the XTS-300 and the generated C code is of questionable origin.

Although Trusted RUBIX and DSA were integrated on the XTS-300, problem #13 prevents successful operation at this time.

3 Proof-of-Concept Deliverable

The actual delivery resulting from this proof-of-concept effort will be Trusted RUBIX integrated with the Datacraft DSA on a Sun SPARC platform, i.e., the integration work completed in August 1996. X.500 functionality will be demonstrable, though the TCB will not be assured, and so the proof-of-concept should not be used in any operational context.

4 Conclusion

The MLS Directory Server proof-of-concept provided a unique opportunity to study the design of an MLS Directories based on the X.500 Series of Recommendations. This proof-of-concept really only touched the surface, however, of what would be required to develop a fully functional, operational-caliber MLS Directory Server.

As a result of this project, Wang and Van Dyke have begun work with the NSA's MISSI program to specify and design such an operational MLS Directory Server which will support true MLS labeling and schema design, MLS replication using DISP shadowing, DSP chaining at multiple security levels, strong two-way authentication and signed operations (as per ACP-133), and an acceptable level of aggregate assurance (based on the highest possible levels of certified assurance of component Trusted Computing Bases).

Based on our discussions with the MISSI program , it seems likely they will task Wang (and Wang-designated team of subcontractors) to develop such an operational MLS Directory Server, most likely in late 1998 or early 1999 time frame, as part of the MISSI Network Security Manager program. At that time, the lessons learned from the Rome Laboratories MLS Directory Server proof-of-concept will be important to help assure the success of the effort.

APPENDIX A

MLS X.500 Directory Server Functional Specification

MLS X.500 Directory Server Functional Specification

Version 1 — 26 February 1996

Prepared for:

UNITED STATES AIR FORCE ROME LABORATORY
RL/C3AB
525 Brooks Road
Rome, NY 13441-4505

Prepared by:

Karen Goertzel
WANG FEDERAL, INC.
7900 Westpark Drive
McLean, VA 22102-4299

Laura Boyer
J.G. VAN DYKE & ASSOCIATES, INC.
141 National Business Parkway — Suite 210
Annapolis Junction, MD 20701

Mark Smith
INFOSYSTEMS TECHNOLOGY, INC.
6411 Ivy Lane
Greenbelt, MD 20770

1.0 INTRODUCTION AND BACKGROUND

The United States Department of Defense (DoD) and allied defense ministries and departments are migrating to the use of open system solutions based on international standards for their messaging, network management, security, and document interchange systems. The U.S. DoD is developing a single messaging system for all individual user and organizational messaging. This Defense Messaging System (DMS) will use the X.400 Message Handling System protocols combined with the Secure Data Network System (SDNS) Message Security Protocol (MSP), the X.500 Directory System protocols, and the Common Management Information Protocol (CMIP). The combination of these technologies will provide the DoD with the required messaging, security, network management, and directory services to implement global messaging capabilities. The X.500 Directory System will provide an integral part of the DMS infrastructure, by providing a means to store and distribute addressing and security information.

The current DMS solution addresses the Sensitive-Unclassified environment. As DMS evolves to address the requirements of SECRET and TOP SECRET environments, the storage, distribution, and maintenance of classified directory information will become a large problem. Our MLS X.500 Directory Server will solve this problem and many others. In June 1995, the Director of Central Intelligence mandated that all intelligence services and agencies (S&As) will use the DMS. These organizations have serious concerns about storing directory information in Sensitive-Unclassified directories. In response to these concerns, the MLS X.500 Directory Server could be used to store, distribute, and maintain information at any security classification level, and at multiple classification levels within the same X.500 directory.

Many other U.S. government S&As are also beginning to use X.500 solutions. These include:

- Internal Revenue Service (IRS)
- Department of Energy (DOE)
- General Services Administration (GSA)
- National Aeronautics and Space Administration (NASA)
- U.S. Postal Service (USPS)

In addition, several allied defense departments/ministries, including those of France, the United Kingdom, and Australia, are implementing their own global messaging systems, and these systems will have very similar requirements to that of the U.S. DMS, including requirements for X.500 Directory Service that can accommodate information different levels of security classification.

This document defines the functional requirements for a multilevel secure (MLS) X.500 Directory Server. This system will use the Infosystems Technologies Inc. (ITI) Trusted RUBIX database to store X.500 directory information at multiple security classification levels. The Trusted RUBIX database will be ported to the Wang Federal Inc. XTS-300 B3 Trusted Computing Base. A commercial-off-the-shelf (COTS) X.500 Directory Server Agent (DSA) developed by Datacraft Technologies Pty. Ltd. is being evaluated to assess its portability to the XTS-300, and its ability to be integrated with the Trusted RUBIX database. This product, or another COTS X.500 DSA, will be used to provide the X.500 communication protocol interface to other X.500 systems.

This functional specification describes how the XTS-300 platform, the Trusted RUBIX database, and the X.500 DSA will be integrated to form an MLS X.500 Directory Server. The initial implementation of this MLS Directory Server has been structured as a research and development proof-of-concept funded by Rome Air Development Center. This is limited to defining the functional requirements that must be supported by the individual COTS components to enable the integration of the MLS Directory Server; it does not deal with general functional requirements of those components.

2.0 OPERATIONAL AND ARCHITECTURAL ASSUMPTIONS

2.1 Architecture and Features of Components

1) The architecture and features of the XTS-300 are described in Appendix A.

2) The architecture and features of the Trusted RUBIX relational database management system are described in Appendix B.

3) The architecture and features of standard X.500 Directory Service are described in Appendix C. When a particular X.500 product is selected for this project, Appendix C will be updated with information about that specific implementation.

2.2 Operational Assumptions

1) Network connections into the directory server will be single-level.

2) External DUAs will be single-level.

3) External DSAs will be single-level.

4) Sessions (for query or update) will be single-level.

5) A user who connects at one security level will be allowed to read all data dominated by that security level.

6) A higher-cleared user who wishes to write (add, modify) data at a lower level than his clearance must connect at the level of the data he wishes to write.

7) A user cannot write data at any level but that of the connection over which he accesses the directory server. There can be no "write up", regardless of Bell-LaPadula policy.

8) *DSA chaining, referrals, and multicasting:* The current proof-of-concept may have limited chaining capability allowing single-level chaining between the MLS Directory Server and an external DSA operating at the level of the DUA that initiates the lookup or update that necessitates the chaining. It is not expected to be able chain to DSAs at lower levels dominated by the level of the initiating DUA, though this capability is proposed as a future enhancement (see 4.5). Nor is it expected to refer or multicast, as these functions are being strongly discouraged by the DMS program.

9) *The Use of Fortezza in the proof-of-concept:* The use of Fortezza in this proof-of-concept will be for strong two-way identification/authentication between external DUAs/DSAs and the MLS Directory Server only. Fortezza will not be used to convey the security level of the communication connection or session. See #8 above.

10) *Determining which DUAs can perform updates:* The Directory Server will be able to determine which DUAs are authorized to update the Directory Information Base (DIB), and which DUAs may only query the DIB. This determination may be based on the DUA's I&A information.

2.3 Architectural Assumptions

1) There will be a single internal DSA at each security level for which data is stored in the multilevel database (DIB).

2) There will be a security level in the DIB that correlates to each single-level internal DSA.

3) External DUAs and DSAs will communicate only with the internal DSA at their own level. They will not be able to communicate with an internal DSA at any other level than their own. To enable an external DUA to look up information at all levels dominated by the DUA's security level, "downgrading" of DAP lookup requests to the internal DSA will be performed as a standard database function of Trusted RUBIX.

4) For the purposes of this proof-of-concept, network connections will be considered single-level, and will be determined by the classification level of the discrete physical network over which the connection is made. The handling of security levels logical connections will be an issue for future study. See Section 4.3.

See Figure 1 for an functional diagram of the Proof-of-Concept MLS X.500 Directory Server.

2.4 Internal Interfaces

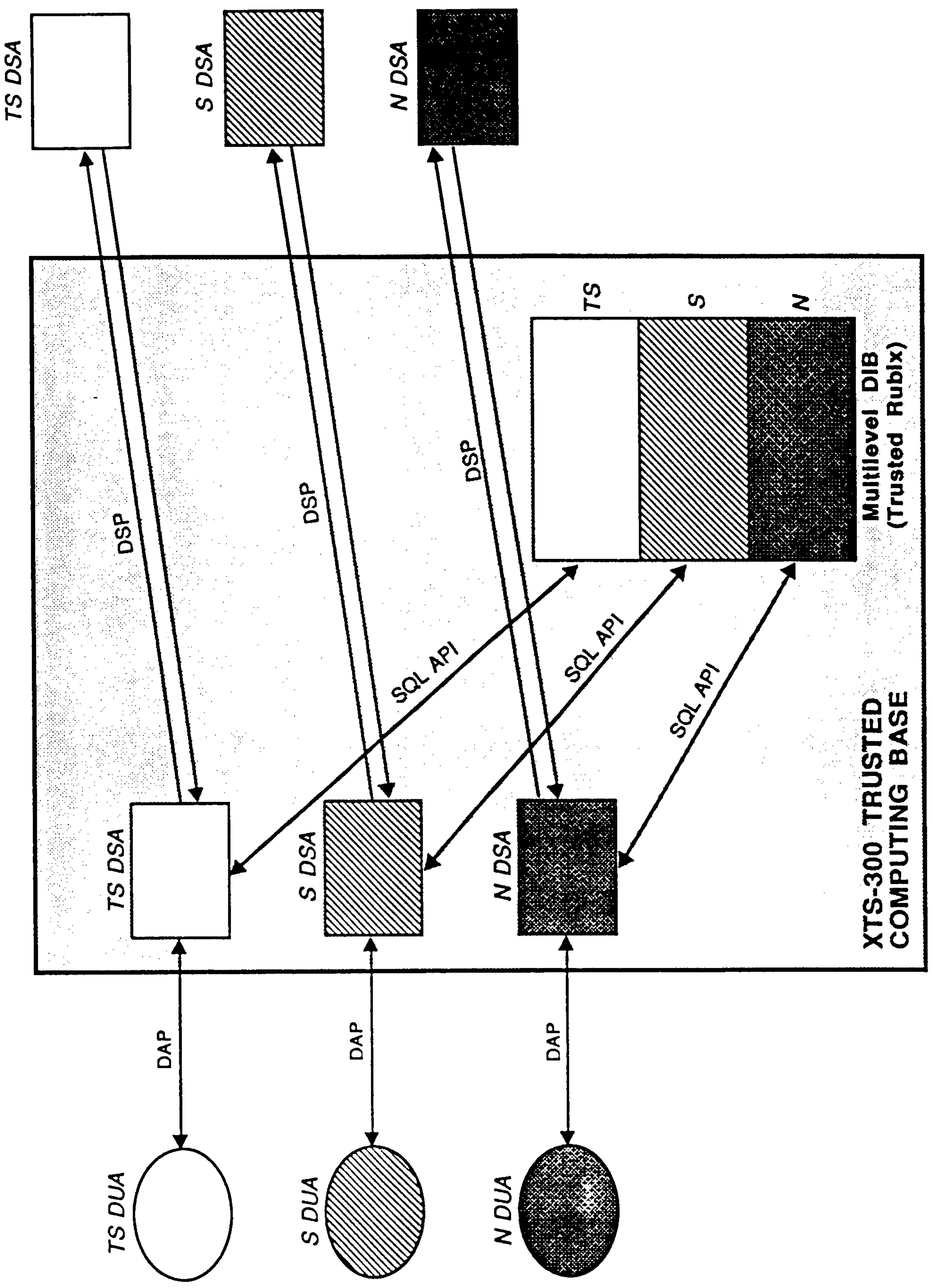
The database access protocol implemented between the DSA and the DIB will be a SQL API.

2.5 External Interfaces

The integrated MLS directory server will appear to all external DUAs, DSAs as a single DSA. The integrated MLS directory server will support all X.500-1993 standard protocols for communication between itself and other X.500 entities, including DAP for DUA-DSA communications, DSP for DSA-DSA chaining, and DISP for actual DSA-DSA shadowing. A proposed future enhancement (see Section 4.5) will be to implement DOP for negotiating DSA-DSA shadowing agreements.

The integrated MLS Directory Server will implement the TCP/IP communications stack through the TCP layer. On top of this will run an RFC 1006 TCP-to-TP4 transition capability. On top of this will run all necessary OSI protocols to enable correct functioning of DUA-DSA and DSA-DSA communications. It is expected that all necessary communications functionality from RFC 1006 on up the OSI stack will be included in the DSA product, and will function with the TCP/IP stack provided on the XTS-300 platform.

Figure 1. **Integrated MLS X.500 Directory Server**
with single-level chaining



3.0 MLS X.500 DIRECTORY SERVER FUNCTIONS

The MLS Directory Server is anticipated to support all X.500-1993 standard DSA functionality provided by the DSA product on which it is based (e.g., Datacraft DX500 OpenDirectory). The integration of this DSA with Trusted RUBIX, and its implementation on the XTS-300, should not limit the X.500 functionality in any way, except in ways constrained by the Mandatory security policy of the system

To the extent possible, configuration of the mandatory policies of the STOP Operating System and the Trusted RUBIX RDBMS will be implemented to minimize impact on the X.500 functionality of the MLS Directory Server.

3.1 Bind Process

When a directory user want to connect to the DSA, to look up or update information, the user's DUA must first connect—or *bind*—to the DSA via the DAP *bind* operation. Variables of the *bind* operation consists are a version number and the user's credentials. These credentials can be ***simple*** or *strong*. DMS and other programs are mandating the use of strong credentials, based on the use of digital signatures generated by public-key cryptosystems. This strong authentication is supported in accordance with the X.509 Authentication Framework. The integrated MLS directory server will make its access control decisions based on the user's authenticated credentials.

3.2 Directory Lookup Access

The integrated MLS directory server must appear to all external X.500 entities (DUAs, DSAs) as a single DSA. However, unlike a single-level DSA, the MLS Directory Server can allow any single-level DUA at any classification level (dominated by the user's clearance level) to look up (database *read*) information not only at its own classification level, but at all classification levels below it. This ability would be enabled by the fact that there will be multiple security levels of data stored in the multilevel database that acts as the directory server's Directory Information Base (DIB), in accordance with and enforcing the Bell-LaPadula model.

For example, if the DIB contained data ranging in classification from UNCLASSIFIED-BUT-SENSITIVE to TOP SECRET, a TOP SECRET DUA would be able to read information at the TOP SECRET, SECRET, CONFIDENTIAL, AND UNCLASSIFIED-BUT-SENSITIVE levels. By contrast, an UNCLASSIFIED-BUT-SENSITIVE DUA would have access to only to UNCLASSIFIED-BUT-SENSITIVE data *within the same DIB*. It is a privileged process in the database management system (Trusted RUBIX) on which the DIB is based that performs the actual reclassification of lower-level data before handing it "up" to the higher-classified user over the single-level session/single-level network.

The Relational Database Management System (RDBMS; Trusted RUBIX), supported by the MLS operating system (STOP), will enable and enforce all multilevel database read-accesses.

3.3 Directory Update Access

Directory update (including add, delete, modify, and modifyDN, i.e., database *write*) operations, however, are constrained by more than the Bell-LaPadula model. These are constrained by the level of the network connection over which the write request is transmitted, and by the level of the requesting DUA. Thus, write operations will only be supported at the same level as the DUA and network connection. For example, a TOP SECRET DUA would only be able to update, add, delete, modify, and modifyDN TOP SECRET data (although that DUA could look up TOP SECRET down to UNCLASSIFIED-BUT-SENSITIVE data); an UNCLASSIFIED-BUT-SENSITIVE DUA would only be able to access UNCLASSIFIED-BUT-SENSITIVE data.

The RDBMS will allow only single-level update accesses, with the level of the access determined by the level of the DSA generating the DSP/SQL chained update request. If the RDBMS receives a DSP call from a higher-level DSA requesting write-access to a lower level in the DIB, it will prevent the access from occurring.

3.4 Access by Other MLS X.500 Entities

While it will not be possible, with this proof-of-concept implementation, to implement a labeling or other mechanism to enable the preservation of knowledge of security labels between our MLS directory server and some other vendor's multilevel X.500 entity (if such exists), it should be possible to maintain knowledge of security levels of data transmitted between MLS directory servers developed by us. See Section 4.9.

3.5 MLS Directory Server Processing

3.5.1 Directory Lookup, All Data Present in Local DIB

1) In this implementation, an external DUA will connect to the MLS Directory Server at the classification level of the DUA platform's physical network connection to the XTS-300.

2) A process within the MLS Directory Server will route the requested *bind* from external DUA to the internal DSA that operates at the same level as the DUA. The *bind* will be accomplished as per the description in Section 3.1 of the X.500 *bind* operation.

3) The internal DSA will perform the necessary processing of the DUA request, including sending a SQL query to Trusted RUBIX to retrieve the requested information from the DIB.

4) Trusted RUBIX will derive the security level of the SQL query from the security level of the requesting DSA (which acts as a Trusted RUBIX client). Trusted RUBIX will then attempt to retrieve the data to satisfy the SQL query, within the limitations of the system's security policy. For example, in response to a TOP SECRET query, Trusted RUBIX will attempt to retrieve data at all levels dominated by TOP SECRET (i.e., TS and below); in response to a Sensitive-Unclassified query, Trusted RUBIX will attempt to retrieve only Sensitive-Unclassified data.

5) If the data required to satisfy the DUA request exist in the Trusted RUBIX DIB, Trusted RUBIX will retrieve them and return them to the requesting internal DSA. If the data required to satisfy the query exist at multiple security levels in the DIB, Trusted RUBIX will perform the necessary privileged operation to raise the classification of lower-classified data to the level of the query, then combine the data into a response that satisfies the query.

6) The internal DSA will send the response (including the requested data) to the external DUA.

7) The external DUA will reply by requesting another lookup, an update, or by unbinding from the internal DSA.

3.5.2 Directory Lookup, Some or All Data Absent from Local DIB

If the data required to satisfy a DUA request do not exist in the local DIB, the following steps replace 3.5.1(5) through 3.5.1(7):

5) Trusted RUBIX will return a "no information available" response to the internal DSA in Step 3.5.1(5).

6) The internal DSA will chain the unfulfilled request, via DSP, to an external DSA operating at the same security level as the internal DSA and the originating DUA.

7) The external DSA will send the requested information (which it will have stored locally, or which it will need to chain to another DSA to retrieve) via DSP to the internal DSA.

8) The internal DSA will:

- *If all requested data were received from the external DSA:* forward the response from the external DSA to the external DUA;
- *If part of the requested data were received from the external DSA, and part existed in the local DIB:* combine data and forward the combined response to the external .

9) The external DUA will reply by requesting another lookup, an update, or by unbinding from the internal DSA.

3.5.3 Update of Directory Information

If the DUA requests an update operation rather than a lookup operation, the following steps replace 3.5.1(4) through 3.5.1(7):

4) Trusted RUBIX will derive the security level of the SQL update from the security level of the requesting DSA (which acts as a Trusted RUBIX client). Trusted RUBIX will then attempt to update the data to satisfy the SQL at the level of the internal DSA ***only***. Unlike lookup operations, where a higher-level DSA can lookup data at levels lower than its own, a higher-level DSA cannot update data any level other than its own; it cannot write down.

5) Trusted RUBIX will update the data in the local DIB, and return an acknowledgment to the internal DSA that the update has been committed. If the data required to satisfy the update request do not exist in the local DIB, the DSA will chain the update request to an external DSA at the same level as the internal DSA (as per the chaining procedure in Section 3.4.2). The external DSA will respond with an acknowledgment that the external DIB update has been committed.

6) The internal DSA will send the response acknowledging the committed update to the external DUA.

7) The external DUA will reply by requesting another lookup, an update, or by unbinding from the internal DSA.

4.0 FUTURES

Based on the success of this proof-of-concept effort, we propose to continue our R&D effort to further enhance the MLS directory server. Our objective will be to transform the proof-of-concept into a desirable system that can meet DMS and other operational requirements. This R&D effort will include, but not be limited to, the following tasks:

4.1 Move Proof-of-Concept to New XTS-300 Release

The July 1996 release of the Wang Federal XTS-300 will run a new version of the STOP operation system, 4.2.1, on a multi-processor Pentium platform. For greatly improved performance, and to enable a re-implementation of the Trusted RUBIX port using the new Ring 2 sockets interface (see 4.2) that will be available in STOP 4.2.1, we propose to recompile/re-link the integrated MLS directory server on the new release of the XTS-300.

4.2 Re-implement Trusted RUBIX Using XTS-300 Ring 2 Sockets

Because Trusted Sockets will not be available on the XTS-300 until July 1996, the proof of concept will be implemented using Ring 2 named pipes for interprocess communications between Ring 2 (trusted) and Ring 3 (untrusted) RUBIX processes. Named pipes, however, present severe performance limitations which, while acceptable for a functional proof-of-concept, will be unacceptable in an operational system. Therefore, we propose to re-implement XTS-300-based Trusted RUBIX to use Ring 2 sockets rather than named pipes for its multilevel interprocess communications.

4.3 Implement Security Labels Based on Logical Connections

For the proof-of-concept, each physical network—and all systems on that network—connected to the XTS-300 is presumed to be operating at single security classification level. The security label to be recognized by the MLS X.500 Directory Server for each external DUA and DSA on a particular physical network will be derived from the level of that network. In future, we propose to derive the security label of a connection from the *logical* communication path of that connection. X.509 Strong Authentication, supported by Fortezza cards, will be used to authenticate and track the security level of the logical connection for the duration of the association between the MLS Directory Server and the external DUA or DSA.

Logical connection-based security labeling will enable the MLS X.500 Directory Server to operate in the future DMS environment, wherein multiple security levels of logical connections, separated by cryptographic means, will be transferred over the same physical network. The use of logical connection-based security labels will be implemented to support chaining and shadowing between the MLS Directory Server and external single-level DSAs. This will include the preservation of the security label for both inbound and outbound DSP, DOP, and DISP communications for the duration of the chaining or shadowing association between the MLS Directory Server and the external DSA(s).

4.4 Implement Multilevel Chaining to External DSAs

In the proof-of-concept, chaining is supported only at the security level of the DUA request which the MLS Directory Server must chain to another DSA to fulfill. This means that to fulfill a SECRET DUA's lookup request, the MLS Directory Server will only chain to a SECRET external DSA, even if the information required by the requesting SECRET DUA actually exists on a lower-level DSA.

To support chaining from the MLS Directory Server to external DSAs operating at multiple security levels, we propose to develop a privileged process to enable the MLS Directory Server to essentially replicate the chaining request to multiple DSAs at each security level dominated by the original DUA. Thus, to fulfill to a SECRET DUA request, the MLS Directory Server would not only chain to an external DSA at the SECRET level, but also to additional external DSAs at the Confidential and Sensitive Unclassified levels, and at the Unclassified level, if security policy permits (see Figure 2). It may be necessary to port a DUA to the XTS-300 to support this multilevel chaining capability.

4.5 Support Two-Way DSA Shadowing Capability

The proof-of-concept MLS Directory Server will support DISP for replication using shadowing. We propose to implement a capability for the MLS Directory Server to authenticate the security level of an external DSA that requests shadowing of the information stored in the MLS Directory Server. Based on this authenticated security level, the MLS Directory Server will limit the information it allows to be shadowed to only information at the security level of the requesting DSA; for example, a SECRET DSA will receive only SECRET information from the MLS Directory Server's multilevel DIB. If the subscribing DSA is also multilevel, multiple associations may be required to distribute information at only the security levels supported by the subscribing DSA. In addition, we propose to demonstrate that the MLS Directory Server can also act as a consumer of information managed and supplied by external single-level and multilevel DSAs, to prove that all updates of directory information do not have to occur on the MLS Directory Server, but that updated information can be shadowed to the MLS Directory Server from external DSAs.

4.6 Implement Full DMS Schema

The directory schema is meta-information that describes the structure and content of the actual information held in the X.500 directory. The schema is built up of layers of definitions, with each layer forming the foundation for the next. Entries in the directory are defined as belonging to one or more object classes, and have various "must contain" and "may contain" attribute types associated with them; each attribute type contains one or more values.

X.500 is flexible in allowing different applications to define a directory schema to support the required Directory Information Base. However, to support a given directory user community, the DSAs supporting that community must be aware of the same schema so they can all process requests for included attributes and entities. The proof-of-concept MLS Directory Server schema will be only a subset of the DMS schema. We propose in future to enhance the MLS Directory Server to support the entire DMS directory schema, as outlined in the DMS X.500 Directory Baseline Schema, 23 February 1996.

4.7 Implement MISSI Protocol Filtering, Dirty Word Scans, etc.

The NSA Multilevel Information System Security Initiative (MISSI) have currently implemented a DAP filter to ensure that any requests leaving a "secure" enclave have been verified to ensure that such requests have first been processed through one or more predetermined filters to strip out "bad" data. It may be necessary to provide filters for DSA and DISP in future. We propose to enhance the MLS Directory Server to make use of these MISSI X.500 protocol filters. This enhancement will bring the MLS Directory Server nearer to compliance with the MISSI requirements for the Secure Network Server-based X.500 guard.

4.8 Implement X.500 Standard Operational Security Enhancements

There is currently under review in the X.500 community a set of Draft Amendments (DAMs) to the X.500 standard support Enhancement of Directory Operational Security. These enhancements include:

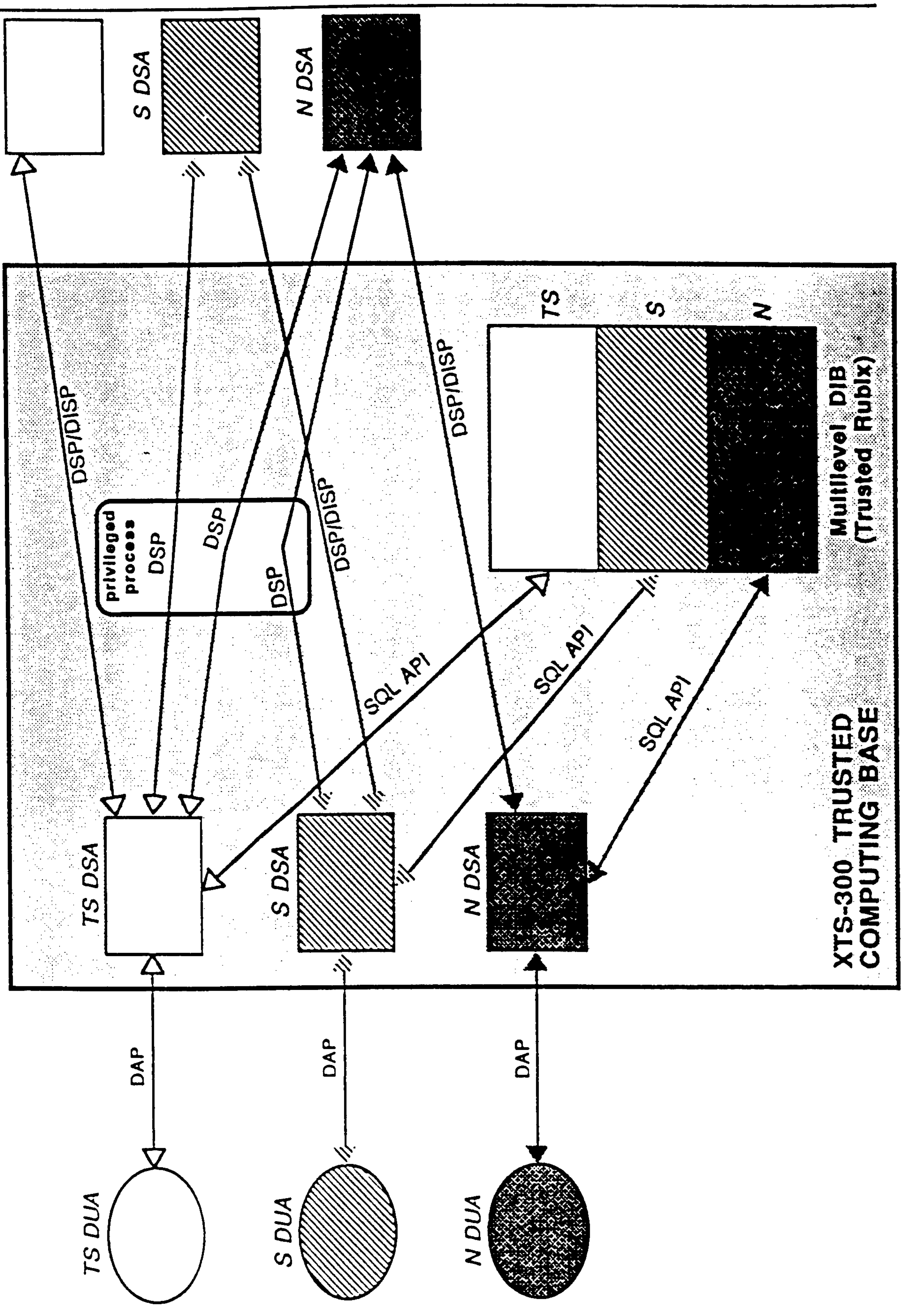


Figure 2. **Integrated MLS X.500 Directory Server**
with multilevel DSA chaining/shadowing

1) Integrity of stored data based on digital signatures;
2) Confidentiality of stored data based on encryption;
3) Auditing facilities;
4) Rules-based access control;
5) Context-based access control.

As these DAMs become stable, we propose to enhance the MLS Directory Server to provide this additional functionality. Such enhancements will not only ensure that the MLS Directory Server remains in compliance with international standards, they may provide the first proof-of-concept in the international X.500 of the practical implementation of these proposed enhancements to the X.500 standard.

4.9 Implement Trusted Labels between MLS X.500 Entities

To enable communications between two MLS Directory Servers, between the MLS Directory Server and another vendor's MLS DSA, or between the MLS Directory Server, and an MLS, we propose implementing a trusted labeling mechanism similar in intent to DNSix and CIPSO.

Appendix A. XTS-300 FEATURES AND ARCHITECTURE

The XTS-300 is described in the following pages.

XTS-300™

B3 Trusted Computing Base

Technical Overview

WANG

XTS-300...the Next Generation

The newest generation in Wang Federal, Inc.'s family of high-assurance systems, the XTS-300™, runs on a commodity 50MHz Intel 80486 EISA bus system. The Intel four-ring chip architecture was selected specifically for its ability to augment the trusted operating system by physically isolating security domains in hardware.

The XTS-300 represents an order of magnitude of improvement in price-performance over the XTS-200. A single-processor XTS-300 delivers up to ***20 times*** the processing speed of a single-CPU XTS-200™. Because it uses commodity hardware, the XTS-300 costs 80% less than the XTS-200.

Running on an Intel™ 486 processor, the XTS-300's STOP operating system is designed to execute programs compatible with the Intel Binary Compatibility Standard 2 (iBCS2) within the constraints of the system's multilevel secure operating model. This means many commercial PC applications can run on the XTS-300, as long as they comply with iBCS2. In addition, increased compliance to the UNIX™ System V Interface Definition (SVID) provides UNIX compatibility to compiled applications.

In today's military, civilian, and commercial environments, information security is critical. Government and private enterprises are concentrating on comprehensive security policies that address all aspects of their business operations, and which use state-of-the-art computer and communications technologies.

XTS-300...a Matter of Trust

A security policy is multifaceted; it must address personnel security (eg, screening and/or granting of security clearances), control of physical access to computing facilities, and data security, ie, the rules for handling sensitive and classified information. A security policy in turn dictates the specific computer and communications security safeguards and countermeasures to be used in a particular information system; such protection mechanisms may include TEMPEST equipment, encryption, and/or a Trusted Computing Base (TCB) designed to authorize users and control access to system resources and data.

The XTS-300 provides a multilevel secure (MLS) Trusted Computing Base combining hardware and software to implement system's security policy. The system represents the culmination of over 20 years' experience and expertise in the development of MLS computing technology. Following Wang Federal's previous-generation XTS-200, which received its Class B3 security certification from the National Security Agency's National Computer Security Center (NCSC) in June 1992, the XTS-300 received its Class B3 level security certification in May 1995.

XTS-200 and XTS-300 are trademarks of Wang Federal, Inc.
UNIX is a trademark of AT&T Bell Laboratories

XTS-300...Protection Mechanisms

The NCSC, in their *Trusted Computer Security Evaluation Criteria* (TCSEC; CSC-STD-001-83)[1], also known as the "Orange Book" (one of several "Rainbow Books" providing guidance on implementing secure, or "trusted" computing environments), define the criteria for determining the level of security—or *trust*—provided by a computer system. The Yellow Book, or *Computer Security Requirements: Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments* (CSC-STD-003-83), and its companion *Technical Rationale Behind CSC-STD-003-85: Computer Security Requirements* (CSC-STD-004-85) describe how to apply the TCSEC guidelines in different modes of operation to protect sensitive and classified information against varying degrees of security risk. In short, the Yellow Book describes the practical application of the TCSEC.

The TCSEC defines four *divisions* into which secure computer systems can be classified according to the security features they provide, and the amount of confidence—or *assurance*— that those security features will operate as documented. TCSEC divisions range from D through A, with D providing the lowest level of assurance and security functionality, and A the highest level of assurance and security functionality. As a Class B3 system, the XTS-300 delivers a very high level of assurance, and the most security functionality possible.

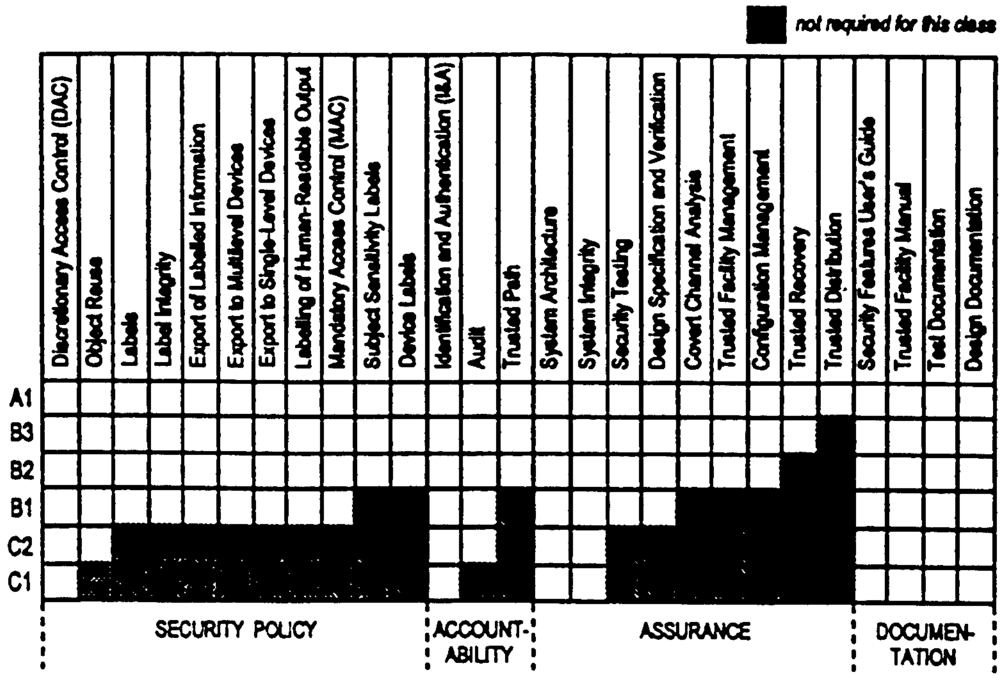
Within each division are up to three *classes*, with each class bringing an increase in security functionality and assurance within the same division. Each division and class is named according to its predominant security feature.

The XTS-300 is a Class B3 MLS system, which means it provides the discretionary security and controlled access protections of Class C2, plus the labelled security and structured protections of Class B2, plus the security domains of Classes B3 and A1. As a Class B3 system, the XTS-300 can simultaneously process and store information at varying sensitivity and security levels in both single-level and compartmented-mode operations, and be simultaneously accessed by users with varying clearances and needs-to-know.

TCSEC SECURITY DIVISIONS AND CLASSES

Division	Classes	Some Evaluated Products
D: Minimal Protection — *sufficient only to protect unclassified, non-sensitive information*	D1	Eyedentify™
	D2	Tigersafe™
C: Discretionary Protection — *considered sufficient for providing user accountability and "need to know" protection defined by the data's owner*	C1: Discretionary Security Protection	IBM RACF™ Ver 1 Rel 5 w/MVS™
	C2: Controlled Access Protection *minimum security for systems procured by US Government.*	Digital Equipment Corporation Open VMS VAX™ Rel. 6.0 • ORACLE 7™
B: Mandatory Protection — *where multilevel security begins*	B1: Labelled Security Protection	SecureWare CMW+™ Ver 1.0 • Hewlett-Packard HP-UX BLS • INFORMIX Online/Secure 4.1
	B2: Structured Protection	Trusted Information Systems Trusted Xenix™ 3.0 • Verdix VSLAN™ 5.0 • Wang Federal Multics™ MR11.0
	B3: Security Domains	Wang Federal XTS-300™ / XTS-200™
A: Verified Protection — *identical in functionality with B3, but provides a higher level of assurance based on extensive documentation and formal proofs*	A1	Boeing Secure Network Server • Wang Federal SCOMP™

SECURITY FUNCTIONS BY TCSEC CLASS



■ not required for this class

	Discretionary Access Control (DAC)	Object Reuse	Labels	Label Integrity	Export of Labelled Information	Export to Multilevel Devices	Export to Single-Level Devices	Labelling of Human-Readable Output	Mandatory Access Control (MAC)	Subject Sensitivity Labels	Device Labels	Identification and Authentication (I&A)	Audit	Trusted Path	System Architecture	System Integrity	Security Testing	Design Specification and Verification	Covert Channel Analysis	Trusted Facility Management	Configuration Management	Trusted Recovery	Trusted Distribution	Security Features User's Guide	Trusted Facility Manual	Test Documentation	Design Documentation
A1																											
B3																							■				
B2																						■	■				
B1										■	■			■					■	■	■	■	■				
C2			■	■	■	■	■	■	■	■	■			■			■	■	■	■	■	■	■				
C1		■	■	■	■	■	■	■	■	■	■		■	■			■	■	■	■	■	■	■				
	SECURITY POLICY											ACCOUNT-ABILITY			ASSURANCE									DOCUMEN-TATION			

XTS-300...Point of Reference

The XTS-300 implements the Reference Monitor concept. A Reference Monitor is the mechanism in an automated information system that enforces permissible, or authorized, access relationships between the system's ***subjects***—ie, active elements that attempt access (ie, processes)—and its ***objects***, ie., passive elements that are accessed (ie, data segments, processes, devices).

Subjects in the XTS-300 can be devices or executing user programs (ie, process-domain pairs). Subjects can be ***trusted*** or ***untrusted***, implying the degree of discipline with which they were developed and with which they operate. Not all subjects need to be trusted to get the job done. Trusted subjects are only used when it's necessary to manipulate the system's high-integrity databases; if necessary, they operate according to controlled exceptions to the customary TCB-enforced access control rules. Thus, a subject is considered trusted only if its integrity level allows it to manipulate TCB databases , or if it possesses privileges that exempt it from specific TCB access control rules.

The Reference Monitor checks every access attempt—or *reference*—against a list of authorized reference types (ie, *read*, *write*, *execute*) allowed to the particular subject attempting the reference. This Reference Monitor check validates that the subject is indeed allowed to make the requested type of reference to the requested object.

The Reference Monitor is an essential element of any computer system that provides MLS processing. For the Reference Monitor of an MLS system to be legitimate, that system's access validation mechanism must be tamper-proof, and must be invoked for *every reference by a subject to an object*. The Reference Monitor is implemented in the system's *Security Kernel*, which uses the specific hardware features of the Intel platform to maintain total isolation between subject and object security levels.

XTS-300...Security and Integrity

The subjects in a MLS system are strictly limited to referencing objects according to the NCSC-approved Bell-LaPadula formal mathematical model of computer security policy[3]. In the XTS-300, this policy is implemented by a set of security rules designed to protect data from unauthorized access. The XTS-300 multilevel TCB implements the Reference Monitor concept and enforces the Bell-LaPadula model, while providing even stricter security * property control. Bell-LaPadula specifies the following mandatory security rules:

- *Simple security:* A subject is allowed to read or execute an object (eg, a data file) only if the security level of the subject dominates (is greater than or equal to) that of the object.

- *Security * property* (read as "Security star property"): A subject is allowed to write an object only if the security level of the object dominates that of the subject. The XTS-300 is even more restrictive in its implementation of security* property protection. It allows the subject to write to an object only if subject and object have the same security level and prevents the problem of lower-level subjects writing higher-level objects that they are then not allowed to read or modify.

The XTS-300 TCB also enforces KJ Biba's integrity policy[3], a corollary to the Bell-LaPadula model that enforces the system's mandatory integrity rules. These integrity rules protect information from unauthorized modification (writing), whereas security rules protect information from unauthorized access (reading). As with its security * property enforcement, the XTS-300 provides even stricter integrity * property control than that called for by Biba. Specifically, Biba integrity policy enforces these rules:

- *Simple integrity:* A subject is allowed to read or execute an object (eg, a data file) only if the integrity level of the object dominates that of the subject.

- *Integrity * property* (read as "Integrity star property"): A subject is allowed to write an object only if the integrity level of the subject dominates that of the object (exception: one process *may* write *up* to another). The XTS-300 goes a step farther, allowing a subject to write an object only if the integrity level of subject and object match.

The XTS-300 TCB supports 16 hierarchical security classifications and 64 mutually-independent security compartments or categories; it also supports eight (8) hierarchical integrity classifications (four for users, one for operating system domain programs, one for operators, one for administrators, and one not assigned) and 16 mutually-independent integrity compartments or categories. The integrity classifications include at least the following three classifications:

```
user < operator < administrator
```

where the subject to the left of "<" is less privileged than the subject to its right.

The XTS-300 TCB includes privileged programs, such as FSM (File System Manager), that allow high-integrity users to circumvent these rules in a highly controlled manner so that they are able to construct a usable file system hierarchy.

The XTS-300 also enforces a discretionary or need-to-know policy, whereby access to an object is determined by the identity of its subjects and/or the groups to which they belong. Thus, the TCB enforces this discretionary access rule:

- *Access modes:* A subject is allowed to access an object in only those mode(s) granted by the owner of the object. Each object shall be assigned allowed permissions (read, write, execute) for the owner of the object, for the members of the owner's group for other specifically identified groups, and for all others.

Each object is referenced by its own unique identifier[3], and each has its own set of access information and status information. This access information includes the object's subtypes and mandatory and discretionary access attributes, and is the basis upon which the Security Kernel makes its decisions. Specifically, an object's *mandatory* access information consists of its security level and categories, and its integrity level and categories.

DISCRETIONARY ACCESS CONTROL

An object's *discretionary* access information includes:

- object's owner and group identifiers;

- read, write, execute permissions for owner, members of groups to which he belongs, and other users;

- up to six (6) user and group identifiers and their specific permissions (read, write, execute);

- brackets specifying rings for read, write, and execute (or control for devices);
- object's subtype (see "Additional Policy Enforcement", below).

The TCB follows a set of general rules to determine whether a subject should be granted discretionary access to an object:

- If subject owns object, use specified owner permissions; *otherwise*
- If entry exists for subject in Access Control List (ACL), use ACL permissions; *otherwise*
- If subject's current group is the same as group of object's owner(s), use specified group permissions; *otherwise*
- If there is an entry for group in ACL, use group permissions; *otherwise*
- If subject has no other specific permissions, use specified "other" ("world") permissions.

ADDITIONAL POLICY ENFORCEMENT

In addition to traditional mandatory and discretionary access rules, the TCB also enforces a user-definable policy that strengthens those rules. This enforcement policy is designed to limit access to objects based on subtype. Each process in the system is assigned one or more accessible subtype lists, one for each type of object in the system. These accessible subtype lists may not be modified by a subject. Nor can the subtype of an existing object be modified (if the object's creator wishes to change its subtype, he must delete the object, then recreate it with the new subtype).

The TCB enforces these access rules for subtypes:

Accessible subtypes: A subject is allowed to access a data object only if the object's subtype is present on the list of subtypes that the subject is allowed to access for that object type.

Object subtype: An object's subtype is specified by the object's creator, and must be derived from the creator's list of subtypes that are accessible for that object type.

XTS-300...the Architecture

The XTS-300's Secure Trusted Operating Program (STOP) is a complete operating system made up of two components: the Trusted Computing Base, that enforces security policy, and the Commodity Application System Services (CASS), a UNIX™ System V Release 3.2-like interface and Intel iBCS2 binary compat-ability, enabling new applications to be easily developed for, and existing UNIX applications to be easily ported to, the XTS-300. These features give system designers flexibility in using untrusted commodity software applications on the XTS-300, while relying on the TCB to provide the security features necessary to yield a high level of security.

Communications and network support for the XTS-300 are provided by the Ethernet-connected Secure Communications Subsystem (SCS), a combination of hardware and software that provides front-end communications processing to the Host Secure Processor (where the TCB runs). The SCS provides the XTS-300's standard commodity network interfaces, which are strictly controlled by the Host Secure Processor. The SCS is hosted on an Intel 486 ISA platform, and supports standard network applications such as TCP/IP, Telnet, and File Transfer Protocol (FTP).

THE XTS-300 OPERATING SYSTEM...STOP

The XTS-300 Secure Trusted Operating Program—STOP—has four primary components:

Security Kernel: small and well-structured to enable complete security evaluation, testing, and verification. The Kernel provides basic operating system services, including resource management, process scheduling, interrupt and trap handling, auditing, and enforcement of the mandatory security policy and discretionary access policy for process and device objects. The security policy is composed of two sets of rules, one governing system security and the other, system integrity.

Trusted System Services (TSS): I/O management, network services, file system management, and enforcement of discretionary access policy for file system objects (ie, services not provided by the Security Kernel) provided to both trusted and untrusted applications and system software. The environment provided by the TSS is controlled by the underlying Security Kernel, which enforces security policy upon the TSS and all other XTS-300 operations.

Trusted Software: includes all security relevant functions that operate as independent services. In some cases, a Trusted Software function may require the ability to bypass the TCB's mandatory and/or discretionary policy controls. For example, trusted processes enable high-integrity users to set up and modify the file system hierarchy to accommodate the use of high-integrity nodes.

Trusted Software functions are available to trusted user processes, system operators, and system administrators,for performing security-related system housekeeping, eg, registering/removing users, assigning passwords, installing and configuring the system, andperforming other operator tasks not supported by the other STOP components. Users can also develop their own trusted functions.

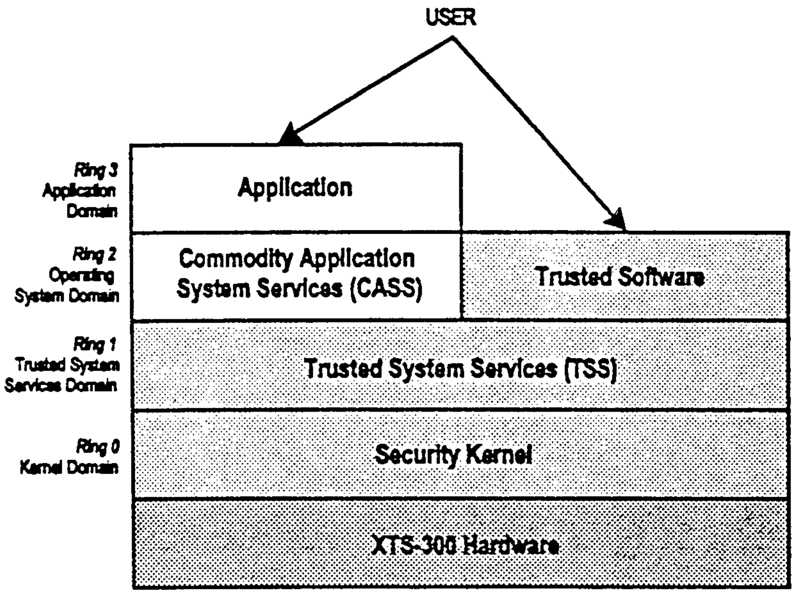
Commodity Application System Services (CASS): an application programmatic interface that provides an implementation of the UNIX System V Interface Definition (SVID), enabling easy porting and development of UNIX applications to the XTS-300. Only those SVID services that violate STOP security policy have been replaced by a CASS equivalent, or eliminated.

RINGS OF ISOLATION

The XTS-300 architecture is built on a hardware ring mechanism which augments the security of the operating system by physically isolating portions of system processes from tampering. The XTS-300 implements four isolated rings, or domains. In the figure, the TCB is represented by the shaded portion.

Ring 0: the most privileged domain; reserved for the Security Kernel which enforces system security policy. I/O device drivers reside in Ring 0.

STOP FOUR-RING ARCHITECTURE


USER
Ring 3 Application Domain
Application
Ring 2 Operating System Domain
Commodity Application System Services (CASS)
Trusted Software
Ring 1 Trusted System Services Domain
Trusted System Services (TSS)
Ring 0 Kernel Domain
Security Kernel
XTS-300 Hardware

Ring 1: reserved for the TSS. Cannot be called or modified by users.

Ring 2: operating system domain; shared between Trusted Software and user-developed trusted processes running in CASS. Trusted Software cannot, however, use CASS features; thus the interface to Trusted Software is proprietary rather than UNIX-like. Ring 2 links the application domain (Ring 3) with the trusted domains of the system. Ring 2 can contain trusted and untrusted software; whether a software process is trusted or not depends on its se-ment, network services, file system management, and enforcement of discretionary access policy for file system objects (ie, services not provided by the Security Kernel) provided to both trusted and untrusted applications and system software. The environment provided by the TSS is controlled by the underlying Security Kernel, which enforces security policy upon the TSS and all other XTS-300 operations.

Trusted Software: includes all security relevant functions that operate as independent services. In some cases, a Trusted Software function may require the ability to bypass the TCB's mandatory and/or discretionary policy controls. For example, trusted processes enable high-integrity users to set up and modify the file system hierarchy to accommodate the use of high-integrity nodes.

Trusted Software functions are available to trusted user processes, system operators, and system administrators,for performing security-related system housekeeping, eg, registering/removing users, assigning passwords, installing and configuring the system, andperforming other operator tasks not supported by the other STOP components. Users can also develop their own trusted functions.

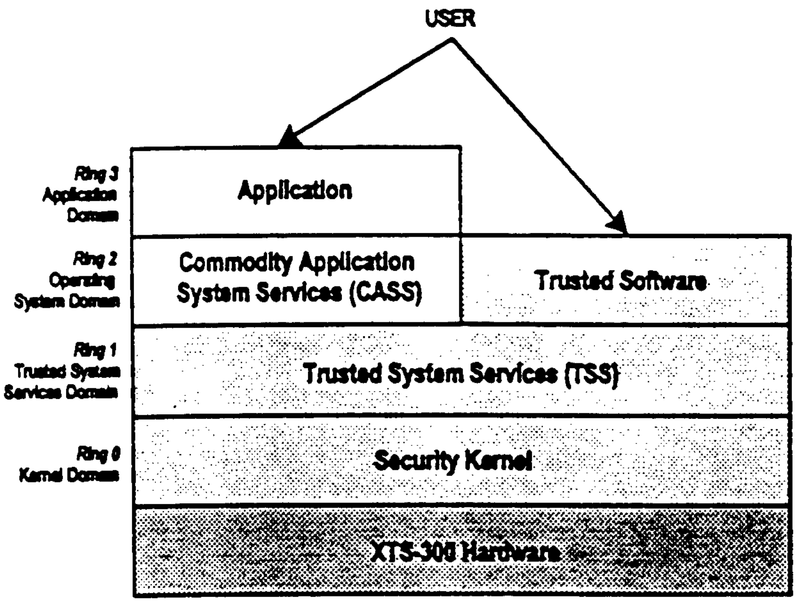
Commodity Application System Services (CASS): an application programmatic interface that provides an implementation of the UNIX System V Interface Definition (SVID), enabling easy porting and development of UNIX applications to the XTS-300. Only those SVID services that violate STOP security policy have been replaced by a CASS equivalent, or eliminated.

RINGS OF ISOLATION

The XTS-300 architecture is built on a hardware ring mechanism which augments the security of the operating system by physically isolating portions of system processes from tampering. The XTS-300 implements four isolated rings, or domains. In the figure, the TCB is represented by the shaded portion.

Ring 0: the most privileged domain; reserved for the Security Kernel which enforces system security policy. I/O device drivers reside in Ring 0.

STOP FOUR-RING ARCHITECTURE


USER
Ring 3 Application Domain
Ring 2 Operating System Domain
Ring 1 Trusted System Services Domain
Ring 0 Kernel Domain
Application
Commodity Application System Services (CASS)
Trusted Software
Trusted System Services (TSS)
Security Kernel
XTS-300 Hardware

Ring 1: reserved for the TSS. Cannot be called or modified by users.

Ring 2: operating system domain; shared between Trusted Software and user-developed trusted processes running in CASS. Trusted Software cannot, however, use CASS features; thus the interface to Trusted Software is proprietary rather than UNIX-like. Ring 2 links the application domain (Ring 3) with the trusted domains of the system. Ring 2 can contain trusted and untrusted software; whether a software process is trusted or not depends on its security requirements, ie, whether it has to update a trusted database, and/or whether it has to be exempt from standard STOP access controls.

Ring 3: application domain, reserved for untrusted user processes. Ring 3 is the users' main programming and processing environment. With no security privileges, Ring 3 is restricted to a local environment for end users; processing in Ring 3 does not affect global system operations or global resources.

Processes in the XTS-300 are subject to Bell-LaPadula and Biba rules bounded by the process isolation enforced by the Security Kernel; that is, processes may access information in a ring of the same or lesser privilege, but not in a ring of greater privilege. All portions of the TCB are protected from unauthorized tampering by one or more mechanisms:

Protection from modification: Security Kernel code and data are protected from modification by any ring other than the Kernel itself. TSS and CASS code and data are protected from modification by processes in any ring other than their own.

Integrity: All TCB program files, databases, and most trusted software processes are protected by setting their integrity level high at an operator's level or higher. Untrusted users (subjects) are excluded from the TCB by restricting their maximum integrity levels in the user authentication database to below that of the TCB objects.

Private segments: Trusted software processes protect their temporary data segments from untrusted software by creating them as *private* segments. Private segments cannot be shared by other processes. This enhances the system's security by isolating the processes from each other.

Secure path: Before a terminal can communicate with the TCB, the operator must strike the Secure Attention Key to disconnect the terminal from an untrusted process. By allowing only one link at a time between the terminal and any Ring 0, Ring 1, or Ring 2 (trusted or untrusted) process, the XTS-300 completely isolates the trusted communications path from the untrusted communications path. (Terminals may be shared by multiple simultaneous untrusted Ring 3 [application] processes.)

Subtypes: An unlocked terminal to be used by trusted software is protected from untrusted software by using terminal-unique device subtypes. When the TCB is entered by establishing the secure path, the secure server removes the terminal subtype from all untrusted processes associated with the terminal session *before* it unlocks the terminal. Access to the terminal is restored to untrusted processes only after the operator exits the TCB.

The trusted databases that contain sensitive user access, group access, session control, and print queue information are protected from unprivileged processes by the use of specific segment subtypes.

TRUSTED DATABASES

Certain information in the XTS-300 is maintained in a protected environment to prevent unauthorized modification. These trusted databases may be manipulated only through the use of *trusted editors*, which are accessible only to users system and security administrators. The XTS-300 trusted databases include:

Audit information database: Contains information about security auditing functions, protected from general access by a subtype. STOP is able to audit up to 256 separate system events, and enables the selective auditing of events, eg, only those events performed by specified users, or only those events occurring above a certain mandatory access level. The security administrator is the only user privileged to modify audit parameters. However, to assure accountability, even he cannot turn off auditing of his own logins and logouts.

Configuration database: Contains system configuration information, eg, timing parameters, site identification, and resource allocation, eg, distribution of shared memory and I/O among multiple processes.

Group access information database: Contains access authentication information for all user groups on the system. Users groups are most often formed to provide a single security profile to a group of users who perform the same business function, have the same mission, or possess the same need to know.

Logical device database: Contains information about each logical device configured on the system, including the devices' access levels. Physical devices: terminals, tape drives, disk drives, etc: are also assigned a logical ID. The system uses this logical ID to manage the device. In this way, the system may segment the total capacity of a single physical hard drive into multiple logical IDs, thus creating multiple logical drives, each of which can be allocated to a different subject.

Printer information database: Contains information on each system printer. System printers are configured to support the marked output requirements of DoD Regulation DoD 5200.1-R. Today's XTS-300 supports simple serial interface (non-laser) printers.

Print request queue: Contains information on each print request.

Security map database: Maps access levels and categories to their I/O character string representation (ie, character strings representing the 16 security classifications and eight integrity classifications and 64 security categories and 16 integrity categories).

Session control database: Contains information to manage each current terminal session.

System directory: A file system directory containing the program images (ie, binary executables) for all trusted programs—including system dæmons—that are not commands. The Kernel and TSS are also stored in the system directory. The system directory's access levels are minimum security and maximum integrity (ie, any user may read, but only the administrator may write).

Terminal configuration database: Contains information about each configured terminal.

Trusted directory: A file system directory containing the program images for all trusted commands (excluding system dæmons, which are stored in the system directory). The access level of this directory is minimum security and maximum integrity (ie, any user may read, but only administrator may write).

Trusted information database: Contains various system parameters, such as start-up flag, log-in banner, default automatic log-out time-out, etc.

User access databases: The user access authentication database contains log-in information, maximum access level, default access level, and privileges for each user. The user access information database contains the user's name, home directory, and his command processor pathname (if any).

XTS-300...MLS Trusted Computing Base

The XTS-300 TCB provides multilevel security (MLS) by simultaneously processing and storing data at different classification/sensitivity levels and need-to-know categories; these multiple levels of data can be simultaneously accessed by users at different clearance levels.

Implementing a MLS TCB can reduce the cost and eliminate the problems inherent with other common approaches to handling multilevel data. The single-level or dedicated mode of operation can require a different computer system to be dedicated to each classification level of data, which is obviously costly; or a single computer may be used, but will require the user to change his own and the system's security level each time he wishes to process data at a different level. Such level changes require cumbersome "scrubbing" techniques before the system can be used to process a different level of data.

The second approach to multilevel data processing requires the use of a "system-high" environment; in system-high mode, all systems and personnel are cleared to the highest level of any information that may be handled in the environment. This alternative is also costly, and results in large numbers of "overly cleared" personnel who don't necessarily have a need to know the information for which they are cleared.

Once information is stored in a system-high system, it assumes the classification level of that system, regardless of its true nature (eg, Not-Classified-but-Sensitive information stored in a SECRET system-high system becomes SECRET, even though the information's content has not changed). For a lower-level user to recover information stored in a system-high system, the "high side" must perform tedious, error-prone manual downgrading to return the information to its original (or appropriate) sensitivity/classification level.

True multilevel secure systems enable the simultaneous storage and access of data classified at different security levels. The range of classification levels that may be processed by a single MLS system is governed by the clearance level of the user and the security evaluation level of that system. The Security Index Matrices in the NCSC *Computer Security Requirements* ("Yellow Book"; CSC-STD-004-85) show which evaluation level should be employed for each possible combination of user clearance level and data sensitivity level in open and closed security environments[5].

WHY B3?

According to the NCSC, a Class B3 system is required for storing and processing of data in open security environments where operating mode is *controlled multilevel* and user clearances and data sensitivity fall within the ranges indicated in the "Security Index Matrix for Open Security Environments", taken from the Yellow Book (CSC-STD-004-85); this matrix is reproduced on page 9, with ranges requiring Class B3 or greater highlighted.

A Class B3 system is also required in closed security environments where operating mode is *multilevel*, and user clearances fall within slightly different ranges (refer to the Yellow Book for its "Security Index Matrix for Closed Security Environments").

"YELLOW BOOK" SECURITY INDEX MATRIX

Maximum sensitivity of data

Minimum clearance or authorization of users

	U	N	C	S	TS	1C	MC
U	C1	B1	B2	B3	NA	NA	NA
N	C1	C2	B2	B2	A1	NA	NA
C	C1	C2	C2	B1	B3	A1	NA
S	C1	C2	C2	C2	B2	B3	A1
TS (BI)	C1	C2	C2	C2	C2	B2	B3
TS (SBI)	C1	C2	C2	C2	C2	B1	B2
1C	C1	C2	C2	C2	C2	C2	B1
MC	C1	C2	C2	C2	C2	C2	C2

A Class B3 system *may* be used in ranges which would normally require A1 systems (or where data exchange would be prohibited) *if* the technical security of the B3 system is augmented by applying additional personnel, physical, or administrative safeguards and countermeasures. This would include higher risk environments where prescribed A1 systems are considered impractical, or where current technology is unable to provide sufficient protection on its own.

XTS-300...MLS in the Real World

To organizations who don't fully understand its benefits and applicability to their mission, MLS can seem excessive, both in terms of management and cost.

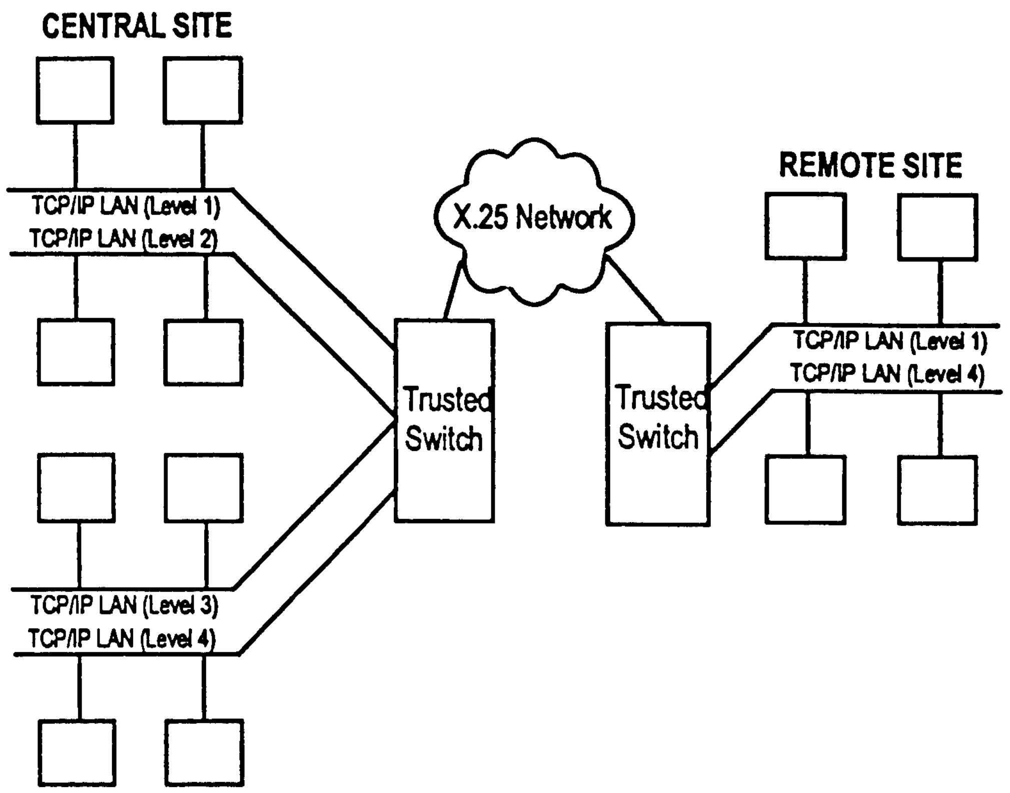
The truth is, the cost of operating in a completely MLS environment is beyond the reach of many organizations. But then, most organizations do not require complete MLS from end to end. Their requirements can be satisfied by hybrid environments composed of "islands" of dedicated-mode or system-high processing (eg, communities of interest) linked together in a highly-controlled manner by Trusted Switches or Guards. With its Class B3 security functionality and assurance, and commodity communications interfaces, the XTS-300 is the ideal platform for high-assurance Trusted Switch and Trusted Guard applications.

XTS-300 AS TRUSTED SWITCH

A Trusted Switch controls access from users' single-level terminals operating at different security levels to multiple host systems operating at different security levels. The Trusted Switch then plays the dual role of guarding access to the host systems and providing a single—or unitary—log-in for each user.

With a Trusted Switch, the security administrator defines the access profile for each user—listing the systems he is allowed to access, the functions he is allowed to perform on each system, the classification levels and categories of information he is allowed to access, and the manner in which he is allowed to process it. The user then has the convenience of a single log-in to the Trusted Switch, which controls all access to the multiple hosts.

TRUSTED SWITCH CONCEPT



Except for log-in, the Trusted Switch acts transparently as the interface between users and the host system(s). Running as a trusted switch, the XTS-300 introduces no noticeable overhead or degradation to end-to-end network performance.

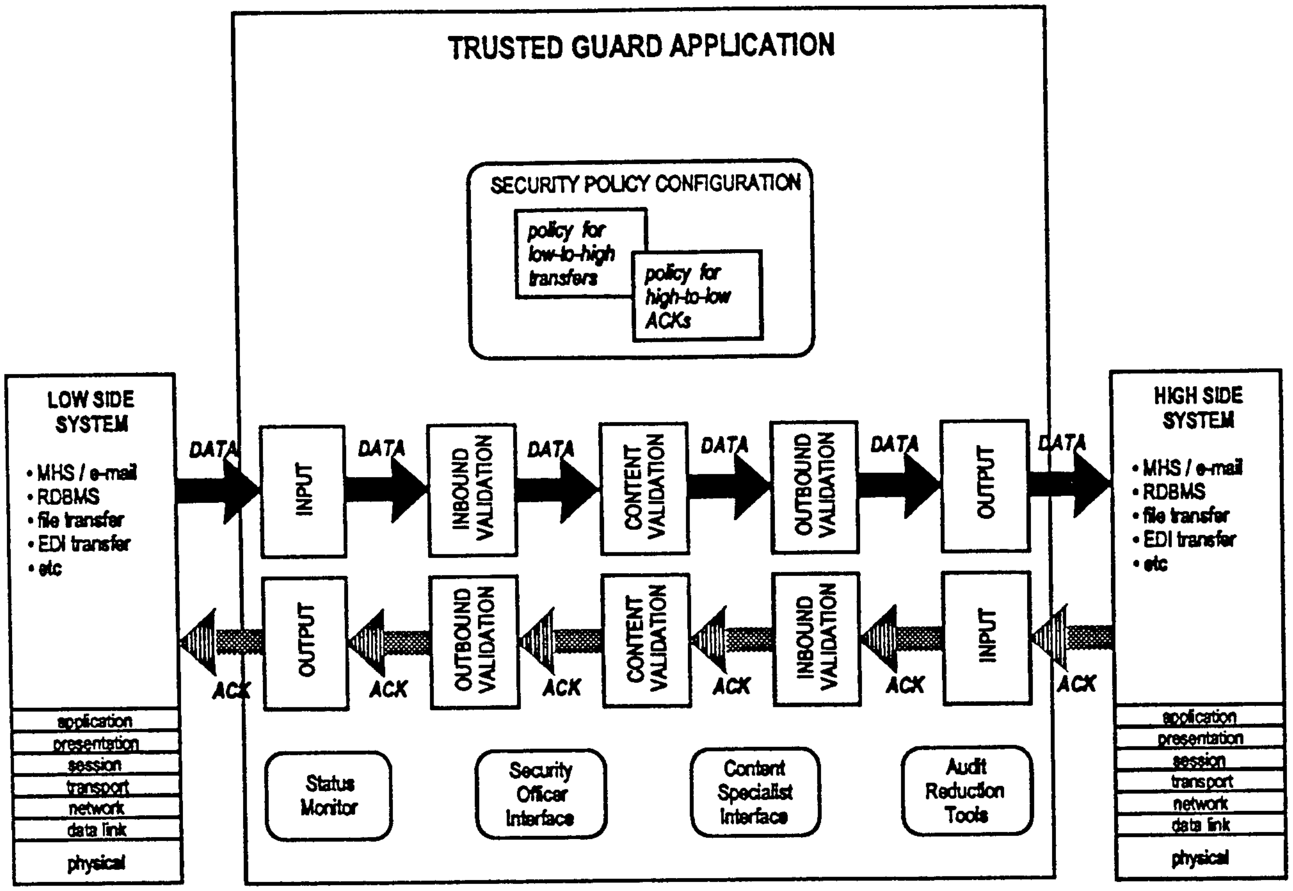
XTS-300 AS TRUSTED GUARD

A Trusted Guard sits between host computer systems operating at different security levels, and controls the transfer of data among the systems. A Guard can (1) validate all requests for information to ensure that the requestor possesses the required access permissions to the desired data; (2) analyze the data's security profile to ensure that this profile complies with the security environment of the receiving system; and (3) perform data processing required by the security policy governing the transfer, eg, classification regrade, sanitization, encryption, auditing.

A Trusted Guard can automate what is currently a tedious, error-prone manual process—the review before transmitting of data stored in a system-high environment to ensure that, even though they are stored in a system at one level, they are really by nature at a lower level, and can thus be downgraded and sent to a system classified at the same level.

Because a Guard automates the scanning and processing of the data, it greatly reduces the margin of error, and vastly speeds the processing of transfers between different level systems. Depending on an organization's security policy, the Guard may be implemented to fully or only partially automate the process; in the former case, the guard might operate according to a set of rules for "sanitization", scanning for "dirty words" (eg, codewords, classified mission names, etc.) in a file, and essentially censoring the file by deleting or replacing those words; or it may simply reject the file transfer and audit the rejection. In the case of a partially-automated Guard, when a file transfer is rejected (ie, the file does not meet the security rules or criteria for multilevel transfer), the Guard may forward it to a human data content specialist for manual processing.

TRUSTED GUARD CONCEPT



References

[1] CSC-STD-001-83, Department of Defense Trusted Computer System Evaluation Criteria, 15 Aug. 1983.

[2] Bell, D.E. and LaPadula, L.J., *Secure Computer System: Unified Exposition and Multics Interpretation*, ESD-TR-75-306; Electronic Systems Division [AFSC], 1976.

[3] Biba, K.J.: *Integrity Considerations for Secure Computer Systems*, MTR-3153; Mitre Corporation, Bedford, MA, April 1977.

[4] An arbitrary number assigned to the object by the Security Kernel when that object is created. The unique identifier remains constant even if the original name of the object is changed.

[5] Open security envrionments are defined as those in which application developers and maintainers are not cleared highly enough (ie, systems processing Confidential and below require programmers cleared to same level as most sensitive data; systems processing SECRET and above require programmers to be cleared to at least SECRET) to provide assurance that they have not introduced malicious logic (eg, Trojan horses) into the system code, and also where configuration control does not provide sufficient assurance that applications are protected against introduction of malicious logic before or during operation.

Closed security environments are those in which developers are sufficiently cleared, and configuration management provides sufficient assurance against introduction of malicious logic.

Wang Federal, Inc.
XTS-300 B3 EPL

Report No.:	CSC-EPL-95/003.A
AS OF:	30 May 1995
MAINTAINED PRODUCT:	XTS-300 release 4.1
ORIGINAL PRODUCT:	XTS-200 release 3.1.E
VENDOR:	Wang Federal, Inc.
EVALUATION CLASS:	B3

PRODUCT DESCRIPTION:

The XTS-300 product is a combination of STOP 4.1, a multilevel secure operating system, and an Intel 80486 PC/AT hardware base using the EISA bus. STOP is a multiprogramming system that can support terminal connections for up to 19 users. Up to 200 processes can run concurrently, each with up to four gigabytes of virtual memory. STOP is designed not only to support much of the UNIX System V interface for applications software, but to produce and run object programs that adhere to a subset of the Intel386 Family Binary Compatibility Specification 2 as well. Network connectivity is provided by the Secure Communications Subsystem, which off-loads lower layer network protocol processing outside the TCB.

STOP consists of four components: the Security Kernel, which operates in the most privileged ring and provides all mandatory, and a portion of the discretionary, access control; the TCB System Services, which operates in the next-most-privileged ring, and implements a hierarchical file system, supports user I/O, and implements the remaining discretionary access control; Trusted Software, which provides the remaining security services and user commands; and Commodity Application System Services (CASS), which operates in a less privileged ring and provides the UNIX-like interface. CASS is not in the TCB.

The XTS-300 uses a standard 32-bit, Intel 80486 PC/AT base with a bus that conforms to the EISA standard. A wide variety of off-the-shelf EISA peripherals are included in the evaluated configuration.

The system provides mandatory access control that allows for both a secrecy and integrity policy. The mandatory security policy enforced by the XTS-300 is based on the Bell and LaPadula security model; the mandatory integrity policy is based on the Biba integrity model. The system implements discretionary access controls and provides for user identification and authentication needed for user ID-based policy enforcement. Individual accountability provided through with an auditing capability. Data scavenging is prevented through object reuse. The trusted path mechanism is provided by the implementation of a Secure Attention Key (SAK).

The separation of administrator and operator roles is enforced using the integrity policy. The system enforces the "principle of least privilege" (i.e., users should have no more authorization than that required to perform their functions) for each of the two defined privileged roles available. All actions performed by privileged users can be audited. The audit log is protected from modification. STOP also provides an alarm mechanism to detect the accumulation of events that indicate an imminent violation of the security policy.

The TCB uses strong architectural characteristics: minimization, layering, abstraction, and data hiding. The TCB makes use of hardware features to provide process separation and TCB isolation and has been designed and implemented to resist penetration. The system design is based on a security model and a descriptive top-level specification.

PRODUCT STATUS:

The STOP operating system was developed by, and is marketed and supported by, Wang Federal, Inc. Release 4.1 of the XTS-300 can be ordered after July 1, 1995. Orders can be placed with:

Robert J. LeBlanc	(703) 827-6910
Technical Projects Manager	FAX: (703) 827-3255
	Internet: leblancr@wangfed.com

Wang Federal, Inc.
7900 Westpark Drive
McLean, Virginia 22102

SECURITY EVALUATION SUMMARY:

The security protection provided by the original product, XTS-200 release 3.1.E, was evaluated by the National Security Agency (NSA) against the requirements specified by the Department of Defense Trusted Computer System Evaluation Criteria [DOD 5200.28-STD] dated December 1985. The NSA evaluation team determined that the system satisfied all the specified requirements of the Criteria at class B3. The original evaluation was completed in May 1992. The original product was produced by HFSI, which has since been acquired by WANG.

XTS-300 release 4.1 is actually based on release 3.2.E of the XTS-200. Release 3.2.E underwent a Ratings Maintenance Phase (RAMP) evaluation based on the original product and successfully completed it in January 1994.

To RAMP release 4.1 of the XTS-300, the vendor maintained the security properties of XTS-200 release 3.2.E, performed configuration management of the changes, and enhanced the security test suite and system documentation appropriately. The security protection provided by XTS-300 release 4.1 has been evaluated against the requirements specified in the Criteria by a joint NSA/vendor analysis team. The NSA/vendor team has also evaluated the system change procedures followed by the vendor against the B2+ RAMP requirements [Eval_Announcements forum transaction 268] dated September 1992.

The joint NSA/vendor evaluation team has determined that release 4.1 of the XTS-300 satisfies all the specified requirements of the Criteria at class B3 and that the vendor satisfied all the B2+ RAMP requirements. The conclusions of the evaluation team have been reviewed and approved by NSA. For a complete description of how XTS-300 release 4.1 satisfies each requirement of the Criteria, see Final Evaluation Report, Wang Federal Inc., XTS-300 (Report CSC-EPL-95/003.A).

The Final Evaluation Report should be consulted for the complete lists of evaluated hardware and software components.

ENVIRONMENTAL STRENGTHS:

The XTS-300 is the only computer system evaluated at class B3 or above (except the XTS-200 and SCOMP, an A1-rated system produced by the vendor which is now out-of-date). The XTS-300 is general-purpose in that it can be used for a range of purposes from personal workstation to multi-user guard/gateway. With additional application support, it is suitable as a network server or firewall. The XTS-300 is a microcomputer based on standard, commodity hardware. At class B3, the XTS-300 provides greater assurance of secure operation than a CMW, B1, or B2 system. It should thereby be accreditable in a wider range of environments than other systems; i.e., it should be accreditable to separate a wider range of data classification levels. Beyond the minimal requirements for a B3 system, the XTS-300 provides a mandatory integrity policy and a familiar, UNIX-like environment for single-level applications. Integrity can be used for virus protection. The UNIX-like environment supports binary compatibility and will run many programs imported from other systems without recompilation.

Appendix B. TRUSTED RUBIX FEATURES AND ARCHITECTURE

Trusted RUBIX is described in the following pages.

Trusted RUBIX

RELATIONAL DATABASE MANAGEMENT SYSTEM

Version 3.0 - March 1996

Infosystems Technology, Inc.
6411 Ivy Lane, Suite 306
Greenbelt, Maryland
(301) 345-7800

OVERVIEW

As government and corporate organizations move to open systems, they require assurance that sensitive data is highly protected against unauthorized access, disclosure, or modification. Trusted RUBIX is the most advanced, secure relational database management systsm (RDBMS) for the UNIX® environment. Only Trusted RUBIX offers the mandatory access controls (MAC) that are part of the highest level of UNIX security — B-3 security as defined by the Department of Defense's "Orange Book." Trusted RUBIX is designed to meet the criteria for B-3 level functionality and assurance.

Trusted RUBIX is configurable in C-2, B-1, B-2, or B-3 level configurations of security. Commercial and government customers will benefit from this flexibility and be able to tailor the product to their own level of security requirements.

Traditionally, information systems have not allowed data to be separated into different sensitivities within a single database. Organizations have been forced to separate data physically on different machines or to provide higher security clearances than necessary for users. Multilevel secure DBMS' minimally allow storage and data processing at different access sensitivity levels on a single machine without risk of compromise. However, many applications, particularly in the intelligence community, require integration or "fusion" of secret data with top secret/special intelligence data. Such applications require B-3 level labeled security protection. RDBMS' targeted to meet only the B-1 level of assurance cannot be used in these types of environments according to government policies. In such environments, the B-3 multilevel security features of Trusted RUBIX are essential. In fact, Trusted RUBIX is the only RDBMS which can provide this level of assurance without giving up any of the traditional functions associated with the products of major RDBMS vendors.

Trusted RUBIX adheres to all critical industry standards in order to deliver unmatched flexibility in deployment, portability of applications and interoperability with other standards-based Open Systems. Users can write to standard Application Programming Interfaces (APIs) such as Embedded SQL (ESQL) and Call Level Interface (CLI). Trusted RUBIX communicates with client workstations and other servers using the Remote Database Access (RDA) standard protocol. Trusted RUBIX is portable across a range of UNIX operating systems.

TRUSTED RUBIX FEATURES

Trusted RUBIX has many advanced features, including the following:

■ A complete set of character string, numeric and date/time data types gives Trusted RUBIX total control over input/output formatting and sophisticated data operations.

■ A client-server architecture which allows untrusted clients to access either a single or multiple trusted servers running Trusted RUBIX.

■ An internal database system design which focuses on the modularity and layering principles which are critical in high assurance systems.

■ ANSI-compliant SQL (X3.35-1992) which also supports many novel language constructs to facilitate data management and security operations.

■ Trusted RUBIX implements ANSI SQL compliant declarative integrity constraints to support entity integrity and referential integrity.

■ Sophisticated techniques for cost-based and heuristic query optimization.

■ A single file RDBMS architecture which supports large, complex databases with an unlimited number of simultaneously open relations, views, and indexes.

■ Through the use of logical views, different users can access and manipulate different portions of the database. The view mechanism is extremely storage efficient.

■ Trusted RUBIX supports both string and binary fields that are varying in length (BLOBs). The maximum field length can be unlimited, subject only to the hardware limitations.

- The centralized database dictionary utilizes the database itself to record the structure of the components which make up the database.

- The unique and sophisticated Trusted RUBIX history mechanism (dated relations) allows access to past data for "what-if" analyses and provides rollback and database concurrency.

- Trusted RUBIX's unique multi-version timestamping concurrency control technique enables the system to securely manage all changes taking place within the database, even with multiple applications running.

- Trusted RUBIX provides discretionary access controls (DAC) which specifies who can do what to the data - who can read, who can insert, who can change, etc.. Only Trusted RUBIX offers the mandatory access controls (MAC) that are part of the highest level of UNIX security available.

- A complete audit trail enables both legitimate (but accidental) errors by users and unauthorized requests to be tracked.

- Supports asynchronous statement execution which means that a statement can be executed whenever the system desires, and control can be returned to the user before execution has completed.

- A savepoint mechanism which allows transactions to be partially rolled back (on user request). Thus, a user can undo all updates performed since the specified savepoint, while at the same time preserving updates performed prior to that point.

- Supports the ability to declare temporary tables which are used only to pass intermediate results from one portion of an application to another. Such tables are "private" to the application that uses them — there is no sharing of data with other applications.

- Trusted RUBIX provides a comprehensive set of tools for monitoring database performance and adjusting resource utilization where indicated. In addition, back-up and recovery facilities ensure your ability to recover database entities on a per relation basis or the database as a whole.

TRUSTED RUBIX COMPONENTS

Trusted RUBIX has four major components:

- SQL
- Embedded SQL
- SQL Call Level Interface (CLI)
- Remote Database Access (RDA)

SQL (Structured Query Language)

Trusted RUBIX fully supports the American National Standard Database Language (SQL) (X3.135-1992). SQL is a widely accepted (by RDBMS vendors and users) non-procedural, English-like query language that is ideally suited for all types of database operations. The set of commands included in SQL are used for a variety of operations including:

- Insert, delete and update functions
- Create, modify, replace and drop functions
- Integrity and consistency features
- Access control features

The SQL syntax and semantics are used for specifying and modifying the structure and the integrity constraints of data and for declaring and invoking operations on the data in a local or remote RDBMS.

SQL and RDBMS are a prime example of the client-server computing architecture which has become a de facto standard. In the client-server model, an application running on one machine can share the task of processing information with another machine, such as a network server.

SQL also allows vendors to create extensions to the standard that enable their products to do tasks that other SQL-compliant RDBMS may not do. A prime area for such extension activity is security or trust.

Embedded SQL

Embedded SQL is a set of database language procedures (SQL commands) embedded within a standard programming language, i.e., C. It includes all SQL commands and additional flow control commands. The embedded SQL statement syntax is compiled (translated) into statements that conform to the particular programming language syntax. Thus, the programmer can enjoy the convenience and notational conciseness of the SQL queries while relying upon the powerful semantics of the C language to control the logic of the application.

Trusted RUBIX embedded SQL may be either static or dynamic. Static SQL is embedded in the procedural language; dynamic SQL usually results when the user writes the SQL statement. A static SQL command embedded in a language like C, is executed as if it were a query entered from the command line. If the programming language cannot handle record level queries, a mechanism called a **cursor** is used to retrieve from a single table.

In general, an SQL query can retrieve many rows (records). The host program (i.e., C) will typically go through the retrieved rows and process them one at a time. The cursor is used to allow row-at-a-time processing by the host program. The **cursor** is like a pointer that points to a single row from the result of a query. The cursor is declared when the SQL command is specified in the program. A cursor command can fetch the query result from the database and sets the cursor to a position before the first row. Each subsequent command moves the cursor to the next row and copies its attribute values into the host program. This is similar to traditional record-at-a-time file processing.

SQL Call Level Interface (CLI)

SQL was originally developed as a way to embed, in an application program, static or dynamic operations on a database. Embedded SQL code is typically converted by an implementation-specific preprocessor into code that is compiled and executed. Dynamic SQL makes SQL more flexible and applies it to cases where the desired database operations are not known at the time the application program is written. Although SQL statements are interpreted dynamically during the course of the program's execution, dynamic SQL is still an embedded invocation technique. As a result, it still typically works through a preprocessor which requires that portable applications be distributed as source code.

The SQL Call Level Interface (CLI) is an application programming interface (API) for database access. CLI is an alternative invocation technique to dynamic SQL that provides essentially equivalent operations. CLI is a set of functions that application programs call directly using normal call facilities. CLI is ideally suited for a client/server environment, in which the target database is not known when the application program is built.

The international standards consortium X/Open, published a Call-Level Interface (CLI) specification which is vendor neutral, platform neutral, and database neutral. Thus, it is possible to use a single API for data definition, manipulation, and access throughout the enterprise. Each application uses the single API to interact with one or several data sources through DBMS-specific drivers. In fact, drivers available from third parties include generic network-communications software, eliminating the need for database-specific protocols. Switching database engines becomes simple — you just change drivers. Porting applications from one platform or database to another can become past history. Trusted RUBIX fully shares X/Open's commitment to "true" open system architectures.

Remote Database Access (RDA)

Organizations are interested in developing means to improve connectivity and interoperability of existing heterogeneous database systems. A significant part of this effort is focused on communication issues involved in accessing data in a distributed computing environment.

Prior levels of interoperability between distributed data sources were limited to homogeneous databases from a single vendor or "gateways" between heterogeneous databases. The end result, was committment to a single database vendor and/or expending an inordinate amount of resources to provide and maintain gateways.

The X/Open international standard Remote Database Access (RDA) specification defines an approach for accessing remote relational data managed on various hardware platforms supported by multiple DBMS vendors. It is based on the use of a standardized SQL application programming interface (API) and the client/server model of computing. The communications service and protocol to permit an RDA server to provide database storage facilities and processing services to RDA clients is part of the specification. The RDA service provides independence such that a user may use the same front end application/development tools to access different DBMS'.

The RDA standard supports two general types of distributed access. The simplest type allows SQL statements within a transaction to access data at a single database server. This type of access uses one-phase commit protocols and is referred to as the RDA Basic Application Context. The other type of distributed access is more flexible since it allows different SQL statements within the same transaction to access different database servers. This more advanced type of access requires two-phase commit protocols and is referred to as RDA Transaction Processing Application Context. Trusted RUBIX provides both types of RDA distributed access.

FEATURE

MVTO

The Cure for Database Locks Disease

CONCURRENCY CONTROL IN TRUSTED RUBIX

Replacing Excessive Database Locking with Multi-Version Timestamping Operations (MVTO)

Concurrency control is one of the fundamental requirements of a database management system (DBMS). Its purpose is to ensure that concurrently executing transactions do not conflict and produce incorrect results. In a multilevel DBMS, the concurrency control mechanisms must also not violate security requirements. In particular, they must not defeat mandatory security or introduce covert channels.

Transactions are conceptual objects that provide the basis for accessing all Trusted RUBIX 2.5 objects. The **transaction** is identified by a **timestamp** (transaction-id) which contains the starting time for the transaction. Due to the requirement that multiple transactions can be processed within a second (the normal UNIX time granularity), timestamps are represented at a granularity of a microsecond. Each transaction terminates by being either committed, which makes the modifications to the relation permanent, or omitted (aborted), which nullifies the changes.

Trusted RUBIX 2.5 provides a mechanism whereby multiple processes can concurrently access and update the same relation at the same time. The outcome of these concurrent actions must be the same as if they were executed one at a time in order of their timestamps. This is referred to as **serializability**. In addition, if a transaction is reading a portion of the relation, it must be able to ensure that the same reads return the same data (i.e., the data is "consistent" and thus does not change during the life of the transaction).

These requirements cause significant overhead and adversely impact database performance. Trusted RUBIX 2.5 has been designed to provide user control over which consistency related problems they will tolerate in order to improve performance.

Consistency problems can be caused by the following phenomena:

Dirty Read

A dirty read is a read operation by a transaction on a record that has been added or deleted by another transaction which has not yet committed its update. If the latter transaction were to omit, the former would be processing data that does not really exist in the database

Non-repeatable Read

A non-repeatable read would occur if a transaction reads the data from a record that is subsequently updated by another transaction.

Phantom Rows

A phantom row would occur if a transaction was allowed to add a record to a relation after a different transaction with a later timestamp had read the relation.

A process can use the **consistency level** to control which phenomena it will allow to occur. As the consistency level gets more restrictive, the performance penalty increases. This should persuade processes to allow the phenomena to occur wherever it does not present problems with the operations of the process.

The consistency level of a transaction which can only be set at transaction start, defines which of the three phenomena are allowed to occur while processing. The following table shows the four consistency levels and defines which phenomena are allowed to occur at each level.

Level	Dirty Read	Non-repeatable Read	Phantom Rows
NONE (0)	OK	OK	OK
MIDDLE (2)	-	OK	OK
HIGH (3)	-	-	OK
VERY HIGH (4)	-	-	-

The Inconvenient and Insecure Way - Locking

The most practiced form of concurrency control – and that which is employed by most commercial databases – is the concept of "locking." Quite simply put, if a data record is being updated by somebody, that data record is "locked" until all the updates are completed. Many database management systems apply locking at the table level. That means, while a given record or set of records in the table is being updated, THE ENTIRE TABLE is out of any other users' grasp. The impact is a detrimental effect on performance as computer cycles are wasted waiting for the table to be available.

Most DBMSs only protect against the dirty read problem noted above. If serializability is desired, the user can specify commands to structure transactions so that serializability is achieved through row-level locking. The remaining phantoms can be avoided by not rereading the same data item, or can be eliminated by setting table-level locks. Either of these methods will ensure serializability, though at a significant cost in concurrency.

Locking has one other major disadvantage: in a multilevel secure system, users can exploit the semantics of locking to convey information to each other in a surreptitious fashion. Suppose that Bob is cleared to see TOP SECRET military information and is able to lock records when he reads them. By locking and unlocking a certain UNCLASSIFIED record at different times, he could send classified data to an unclassified user by the bit patterns generated by the unclassified users' successful and unsuccessful attempts to update a record that was being locked and unlocked.

The Convenient and Secure Way - MVTO

Trusted RUBIX 2.5 uses a concurrency control scheme called "multi-version timestamp ordering," or MVTO. That's just fancy terminology which means that when a record is updated, the old version of the record, which records the values that it had just before the update, remains and can be consulted in case of a mishap. When you start a TRANSACTION, that is, open the database for updating, you are granted a "timestamp," which is a value (taken to be the clock time at the precise moment when you open the database) that is attached to all the records that you update. When you update a record, then, the old value is retired "as of" your timestamp, and the new value is instantiated "as of" your timestamp. This means, you have available what the record looked like before it was changed.

In addition to avoiding locking, timestamps guarantee the concept of "serializability." What that means is that when you have a group of users doing different things to the database in random fashion, the results are EXACTLY THE SAME as if each user came along IN HIS TURN and performed his COMPLETE set of actions on the database. In order to guarantee serializability, certain operations must be disallowed. For example, if I update a record and you come along later and try to update the old version before I've made my changes permanent, your update will be disallowed. Similarly, if you read a record and then later I tried to update it, my update will be disallowed because it will invalidate any calculations that you've performed based upon the old value which you presumed to be the latest value.

Trusted RUBIX 2.5 includes several mechanisms in its implementation of MVTO which capture attempts of multiple processes to update the same record at the same time and attempts to read partial updates that have not been completed. Trusted RUBIX 2.5 also provides for some mechanisms to bypass and/or enhance the way MVTO is implemented on a transaction by transaction basis. These mechanisms provide for several access modes which enable processes to:

- RELAX the serializability constraints at run-time, thereby enabling a process to make the decision between performance vs. serializability (higher serializabilty causes lower performance because of the requirement for additional overhead of ensuring that serializability is maintained). For example, if a user requires full serializability, she can set the consistency level to very high, thereby ensuring that she experiences no phenomena that could affect serializability. However, she could increase concurrent access by setting the consistency level to a lower value, such as high, which would only have minimal effect on serializability, while dramatically improving concurrent access.

- SPECIFY the timestamp of the current transaction and therefore obtain a historical as of view of the database. This is only permitted on read operations.

- RELAX the constraints on the visibility of work performed by other transactions that have not yet completed.

- NOT automatically abort a transaction when an exception is raised. Instead, Trusted RUBIX 2.5 refuses to perform the operation and gives the user the choice of continuing on with other work, or aborting the transaction. This is much more suitable because only the process can know whether or not it must abort because of the exception.

Infosystems Technology, Inc.
6411 Ivy Lane, Suite 306
Greenbelt, Maryland 20770
(301) 345-7800

Appendix C. X.500 FEATURES AND ARCHITECTURE

The X.500 Directory system provides the model, procedures, and means to access a potentially global repository of information. The 1993 version of the X.500 Recommendations provides a robust directory service by defining mechanisms to support distributed environments, with access control on the information that is stored in the directory.

X.500 comprises the following components: Directory User Agents (DUAs) and Directory System Agents (DSAs). In addition, DMS have defined a special component, the Administrative Directory User Agent (ADUA), which will provide authorized users with the ability to maintain the information in the directory via the *add*, *delete*, and *modify entry* operations of the X.500 directory service protocols. Only through the ADUA can the DMS directory be modified; users not authorized for the ADUA will be limited to the standard DUA, which will support only X.500 *lookup* functions, and will not be able to modify the directory in any way.

To support the communications among the DUAs and DSAs, several communications protocols are defined in X.500, including the Directory Access Protocol (DUA-DSA), the Directory System Protocol (DSA- DSA), the Directory Operational (Binding Management) Protocol (supplier DSA-consumer DSA), and the Directory Information Shadowing Protocol (supplier DSA-consumer DSA).

C.1 DUA

The Directory User Agent is a software application that acts on behalf of a user, accessing information stored in the DSA's Directory Information Base (DIB). The DUA connects—*binds*—to the DSA, then performs browse or lookup operations on the information in the DIB. These browse and lookup operations include: read, list, search, abandon, and compare. Directory requests are always initiated by the DUA, which receives results from the DSA. The communications protocol between the DUA and the DSA is the Directory Access Protocol, DAP.

C.2 DSA

The Directory System Agent is responsible for maintaining the information in the Directory Information Base. This information may be distributed and managed by multiple DSAs, which collaborate to gather information in response to a DUA request. The communication protocol for these distributed operations among DSAs is the Directory System Protocol, DSP.

Multiple DSAs combine to provide a global service to users via the shadowing (replication) of directory information. Shadowing provides for data redundancy, faster handling of requests (meeting speed of service requirements), and simplification of directory data management. Shadowing allows directory information to be copied and automatically synchronized from one DSA to another. DSAs which hold entries ("shadow suppliers") can copy these entries to other DSAs ("shadow consumers"), with administrators controlling the frequency and extent of such updates. Updates of shadow directory information may be initiated as a result of a change to the information in the supplier DIB, or it may be performed periodically. Updates may be incremental (i.e., writing only changes since the last update), or total (complete rewrite of the database). Unscheduled updates may also be requested, e.g., for recovery from a system or network outage.

The agreements for managing shadowing can be supported by the Directory Operational (Binding Management) Protocol—DOP—or by bilateral proprietary. The transfer of information to be shadowed is supported by the Directory Information Shadowing Protocol, DISP.

C.3 Directory Access Protocol

DAP defines the exchanges of requests and outcomes between a DUA and a DSA. DAP provides access to the information in the DIB. When a user wants to access this information, the user's

DUA sends a DAP *bind* operation to the DSA. Once connected, the user can read, list, search, and compare the directory information, or abandon his browse/lookup request. If the user has special permissions, he can also add, delete, and modify entries, and modify the Distinguished Name (DN) (in DMS, only ADUA users will have these update permissions). If the DSA can fulfill the request, it returns a DAP result to the DUA.

C.4 Directory System Protocol

DSP defines the exchange of requests and outcomes between two DSAs. DSP provides connectivity between DSAs operating in a distributed environment. DSP has similar functionality to DAP, in that it carries requests and responses to and from the user. Chaining occurs when a DUA makes a request of its local DSA, and the local DSA, because it does not have the information requested, must connect to a remote DSA to fulfill the request. If the remote DSA can fulfill the request, the result is sent back to the DSA, which forwards it to the DUA. In some cases, the remote DSA may have to further chain to another "back-end" DSA, also using DSP.

C.5 Directory Information Shadowing Protocol

DISP defines the exchange of replicated information between two DSAs which have established shadowing agreements via DOP or an agreed proprietary protocol.

C.6 Directory Operational (Binding Management) Protocol

DOP will define the exchange of administrative information between two DSAs that wish to set up a shadowing agreement, to administer operational binding between them. The currently available COTS X.500 DSAs do not yet support DOP. Because of this, the process of negotiating a shadowing agreement is now handled bilaterally using proprietary (non-standard) methods.

APPENDIX B

MLS X.500 Directory Server System Design Document

MLS X.500 Directory Server System Design Document

Version 1 — 23 August 1996
FS96-274-00

Prepared for:

UNITED STATES AIR FORCE ROME LABORATORY
RL/C3AB
525 Brooks Road
Rome, NY 13441-4505

Prepared by:

Karen Goertzel
WANG FEDERAL, INC.

Laura Boyer
J.G. VAN DYKE & ASSOCIATES, INC.

Mark Smith
INFOSYSTEMS TECHNOLOGY, INC.

Rick Harvey
DATACRAFT TECHNOLOGIES PTY LTD

1. SCOPE

1.1 IDENTIFICATION

This document, the System Design Document, describes the detailed architecture and operational environment of the RADC-funded MLS X.500 Directory Server (proof-of-concept phase), hereafter referred to as the MLS Directory Server. The architecture of the MLS Directory Server is based on the requirements identified in the *MLS X.500 Directory Server Functional Specification,* Version 2 (27 June 1996).

1.2 SYSTEM OVERVIEW

1.2.1 Functional Overview

The MLS Directory Server is, at a functional level, a multilevel secure implementation of an X.500 Directory Server Agent (DSA). It enables the storage of multiple classification levels of X.500 directory information in a single Directory Information Base (DIB) managed by a single Directory Server. It strictly enforces Bell-LaPadula security policy rules in the handling of Directory lookup requests, allowing *lookup* access to directory information DUA/DSA users whose clearances dominate the classification level of the Directory information being requested. Unlike strict Bell-LaPadula, however, it enforces a same-level-only *update* policy, allowing an ADUA user to update only Directory information at the same classification level as his clearance level.

When the MLS Directory Server cannot satisfy a lookup or update request, it will chain that request to an external DSA only at the same level as the clearance of the DUA/DSA user whose request the MLS Directory Server could not satisfy. In the proof-of-concept, there is no privileged upgrader/downgrader process that would allow replication of chained requests to external DUAs/DSAs at other security levels (though this kind of trusted process is a key feature of the proposed future enhancement of the proof-of-concept).

This proof-of-concept does not intend to address the operational security policy problem presented by shadowing to/from the MLS Directory Server. X.500 shadowing will be supported in the proof-of-concept, to demonstrate the options for its use. For shadowing *from* the MLS Directory Server, we will demonstrate two shadowing options. In one scenario, we will shadow all levels of data in the MLS DIB (i.e, SBU and Secret) to an external DSA classified Secret, in essence, upgrading the SBU data to "Secret, system high". In a second scenario, we will shadow only Secret data from the MLS Directory to an external Secret DSA, and only SBU data to an external SBU DSA. Information shadowed *to* the MLS Directory Server will be stored in the MLS DIB at the level of the source system. Obviously, in an operational environment, an operational policy decision would have to be taken as to how shadowing would be handled between the MLS Directory Server and external single-level DSAs.

Before any requests can be transmitted from an external DUA or DSA to the MLS Directory Server, an authenticated bind must be established between the two. The authentication of this bind will involve the exchange and authentication of FORTEZZA-generated X.509 authentication information unique to the DUA or DSA it represents.

1.2.2 Architectural Overview

The MLS Directory Server is, at an architectural level, the integration of three software components running on the XTS-300 Trusted Computer System. These components are:

1) One or more copies of the Datacraft DX500 Open Directory DSA;
2) One copy of the Trusted RUBIX MLS RDBMS;
3) FORTEZZA-based Identification and Authentication (I&A) libraries.

All external DUAs/DSAs will see the MLS Directory Server as a single DSA. However, in fact, Directory Server will actually comprise one or more physical DSAs, each at a different classification level. Thus, each external DUA/DSA will actually bind to the internal DSA at its same level. The XTS-300 STOP operating system will enforce the mandatory separation between internal DSAs, and restrict each DSA to communicating only over the physical network interface (Ethernet) at its own level. External DUAs/DSAs are presumed to be connected to system-high (single-level) networks, and each network will connect only to the XTS-300 network interface at the same level, so the internal DSA will only be able to connect over its same-level network interface to external DUAs/DSAs at the same level.

Deriving security labels from the physical network interface level is an understood restriction that will apply only to this proof-of-concept. In future, we intend to derive security labels from FORTEZZA-based X.509 or other trustworthy logical (rather than physical) network authentication information.

1.3 Document Overview

This System Design Document contains the highest level design information for the MLS Directory Server. The SDD describes the allocation of system requirements to HWCIs, CSCIs, and manual operations. The SDD describes the characteristics of each HWCI and CSCI, and is used for two primary purposes: (1) to present the system design to RADC; (2) to provide the design information that will serve as the basis for integrating the MLS Directory Server components. Additionally, the SDD provides an overview of the system that can be presented to other organizations interested in participating in or funding the ongoing development of the "full-blown" MLS Directory Server after the completion of the proof-of-concept phase.

The MLS Directory Server SDD contains the following sections:

- Section 1, SCOPE, identifies the system and provides a system and document overview.
- Section 2, REFERENCED DOCUMENTS, provides a list of all documents referenced in this document.
- Section 3, OPERATIONAL CONCEPTS, describes the mission and operational concepts of the system.
- Section 4, SYSTEM DESIGN, identifies each HWCI, CSCI, and manual operation of the system, and describes the relationship of these items within the system, and the system's external interfaces with other systems.
- Section 5, PROCESSING RESOURCES, describes the processing resources of the system and each configuration item that uses those resources.

2. REFERENCED DOCUMENTS

The following documents were used as resources in the preparation of this document.

- *MLS X.500 Directory Server Functional Specification, Version* 2 (27 June, 1996); Prepared by Wang Federal, Inc., J.G. Van Dyke and Associates, Inc., and Infosystems Technology, Inc., for U.S. Air Force Rome Laboratory/C3AB
- *TCB Subset DBMS Architecture Project: Final Report* (Document Number TR-9404-00-03); Prepared by Infosystems Technology, Inc., for U.S. Air Force Rome Laboratory/C3AB
- *White Paper: Datacraft's DX500 OpenDirectory™*; Published by Datacraft Technologies Pty. Ltd.
- *Technical Brief: DX500 OpenDirectory™ Server Version 3.0;* Published by Datacraft Technologies Pty. Ltd.
- *XTS-300™ Trusted Computing Base Technical Overview, Version* 2 (June 1996); Published by Wang Federal, Inc.

3. OPERATIONAL CONCEPTS

3.1 MISSION

3.1.1 User Needs

The U.S. Department of Defense (DoD) and allied defense ministries and departments are migrating their Information Technology (IT) infrastructures to open system solutions based on international standards for their messaging, network management, security, and document interchange systems. The U.S. DoD is developing a single messaging system for all individual user and organizational messaging. This Defense Messaging System (DMS) will use the X.400 Message Handling System protocols combined with the Secure Data Network System (SDNS) Message Security Protocol (MSP), the X.500 Directory System protocols, and the Common Management Information Protocol (CMIP). The combination of these technologies will provide the DoD with the required messaging, security, network management, and directory services to implement global messaging capabilities. The X.500 Directory System will provide an integral part of the DMS infrastructure, by providing a means to store and distribute addressing and security information.

The current DMS solution addresses the Sensitive-Unclassified environment. As DMS evolves to address the requirements of SECRET and TOP SECRET environments, the storage, distribution, and maintenance of classified directory information will become a large problem. Our MLS X.500 Directory Server will solve this problem and many others. In June 1995, the Director of Central Intelligence mandated that all intelligence services and agencies (S&As) will use the DMS, and the State Department will also be a DMS user. These organizations have serious concerns about storing directory information in Sensitive-Unclassified directories. In response to these concerns, the MLS X.500 Directory Server could be used to store, distribute, and maintain information at any security classification level, and at multiple classification levels within the same X.500 directory.

Several allied government departments/ministries, including those of France, the U.K., and Australia, are implementing their own global messaging systems, and these systems will have data classification policies, and directory and security requirements similar to the DMS's.

3.1.2 Primary Mission

The DMS will begin operation of both Sensitive-but-Unclassified and Secret enclaves by 1 October 1996. The result will be the need to process X.500 directory information at both classification levels. The current DMS plan is to maintain single-level X.500 directories at SBU and Secret, with no exchange of information between them. Unclassified directory information needed by users in the Secret enclave will have to be replicated to the Secret directory server, with an implicit upgrade of the information to Secret. This means that any updates of directory information by Secret users, even though the information originated from the Unclassified enclave, will not be accessible to Unclassified users.

The DMS intends to solve this problem by implementing MLS X.500 guards to enable the transfer of X.500 lookup and update requests between enclaves. However, a more logical approach, in our opinion, would be to place MLS X.500 Directory Servers at the borders of those enclaves, to allow SBU and Secret users to both store and retrieve directory information from a single directory server. This approach will not only eliminate the need for two levels of directory server (reducing the number of overall directories required), it will eliminate the need for MLS X.500 guards — in both cases, reducing the overall cost of deploying X.500 directory service in the DMS environment.

3.1.3 Secondary Mission

A second requirement in the DMS—through the Multilevel Information Systems Security Initiative (MISSI)—is to provide X.509 certificate servers to store, manage, and disseminate the X.509 permission certificates needed for DMS distributed processing. These certificates will be needed by users in the Unclassified and Secret enclaves, and due to the nature of their mission, should be protected at a higher level of security than the non-security sensitive operational components of the DMS (MTAs, UAs, etc.). The MLS Directory Server is perfectly positioned to perform the X.509 Certificate Server function of the DMS, once identified enhancements are put in place (see Section 4.5, "System Design Decisions").

3.2 Operational Environment

The MLS Directory Server in the proof-of-concept phase will be demonstrated in a networked environment designed, at a very basic level, to simulate the dual-enclave DMS:

- System-high Secret network hosting one (1) Secret DUA and one (1) Secret DSA;
- Sensitive-but-Unclassified (SBU) network hosting one (1) SBU DUA and one (1) SBU DSA.

Figure 3-1 illustrates the proof-of-concept network environment.

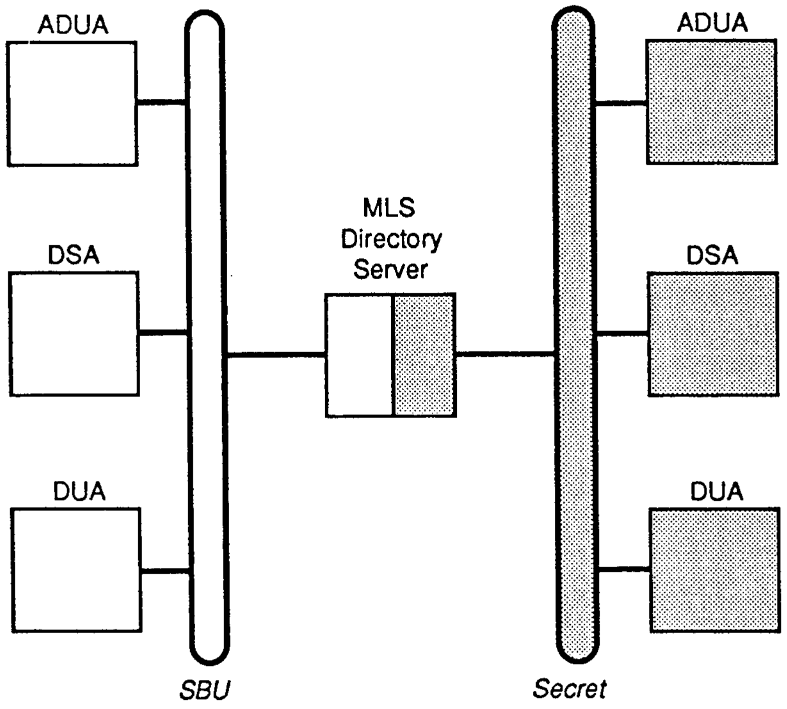


Figure 3-1. Proof-of-Concept Network Architecture

3.3 MLS Directory Server Components and Data Flows

Figure 3.2 illustrates the internal and external components and data flows of the proof-of-concept MLS Directory Server. In the proof-of-concept, there will be external systems, networks, and data at two different classification levels, SBU and SECRET. This limitation is caused by the current XTS-300 restriction that allows it to support only one dual-card FORTEZZA reader, with each FORTEZZA card at different levels. Future XTS-300 releases will support at least one more FORTEZZA reader, enabling support of up to four classification levels by one MLS Directory Server.

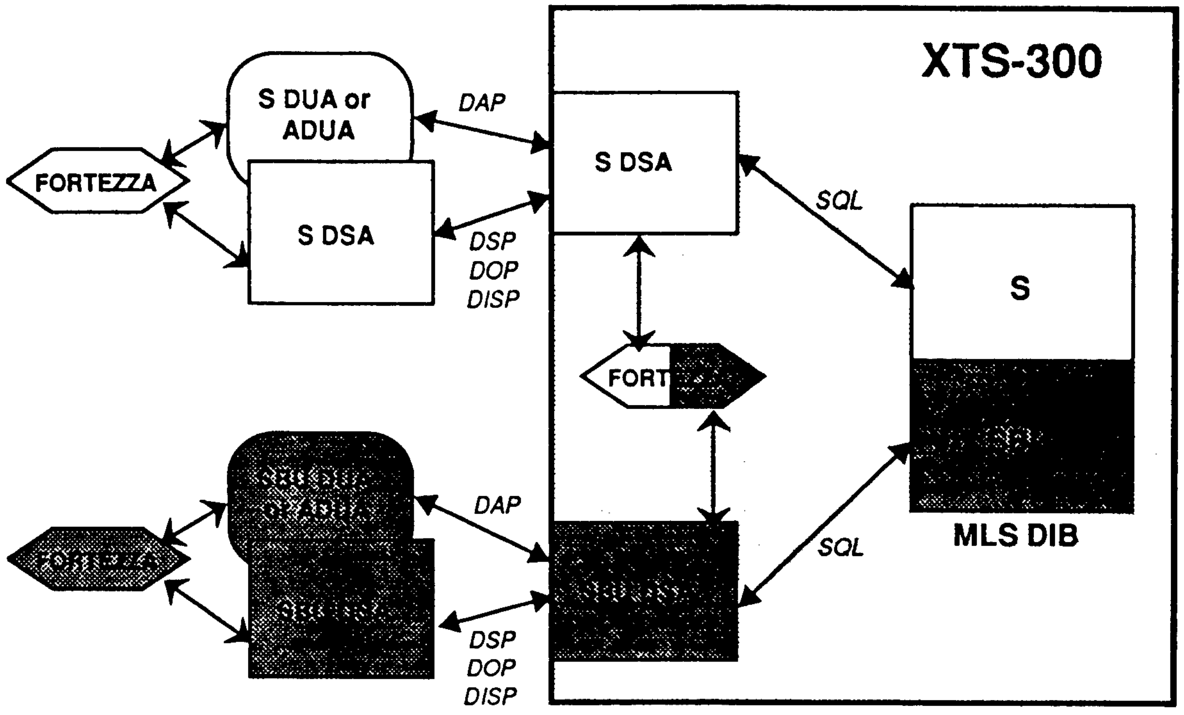


Figure 3.2. MLS Directory Server Components and Data Flows

4. SYSTEM ARCHITECTURE

4.1 Hardware Architecture

The MLS Directory Server proof-of-concept is targeted to run on the XTS-300 Intel 486-based and Pentium-based systems. Figure 4-1 illustrates the two XTS-300 systems.

The hardware architecture for the proof-of-concept is designed to simulate, at a very basic level, the DMS X.500 operational environment with SBU and Secret enclaves. In the proof-of-concept, each enclave will be represented by a single LAN at the appropriate security level, with each LAN hosting a DSA, DUA, and ADUA at the same level. The MLS Directory Server will provide two network connections, one to the SBU LAN and another to the Secret LAN, with security separation between the LANs enforced by the TCB of the XTS-300 STOP Operating System which runs the MLS Directory Server. All components in this proof-of-concept architecture will be equipped with FORTEZZA card readers.

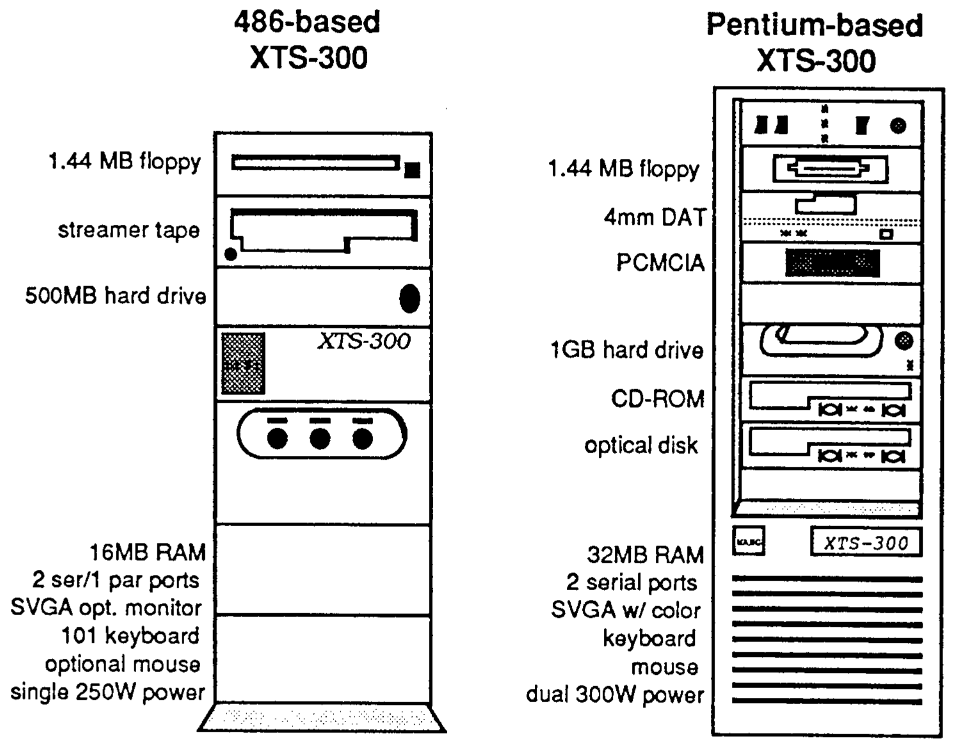


Figure 4-1. XTS-300 486- and Pentium-based Systems

Communications and network support for the new XTS-300 no longer relies on the Secure Communications Subsystem (SCS), a combination of hardware and software that provided front-end communications processing to the Host Secure Processor (where the TCB runs). With the Pentium-based system, the Host Secure Processor itself hosts the Ethernet cards (up to four) and TCP/IP stack and application (FTP, Telnet, SMTP).

4.2 Software Components

The functional components of the MLS Directory Server can be assigned to the following computer software configuration items (CSCIs): Operating System, X.500 Directory Server Agent, Trusted (MLS) Relational Database Management System (TRDBMS), FORTEZZA software, and Configuration Utilities. The major components of these CSCIs are depicted in Figure 4-2.

With the exception of the FORTEZZA I&A libraries, all CSCIs are non-developmental, COTS software products. The FORTEZZA I&A capability will be provided by existing software developed by J.G. Van Dyke & Associates. The objective of this proof-of-concept is to integrate existing software components with a minimum of custom development work. This said, there are some slight modifications that will be made to some COTS components to enable integration.

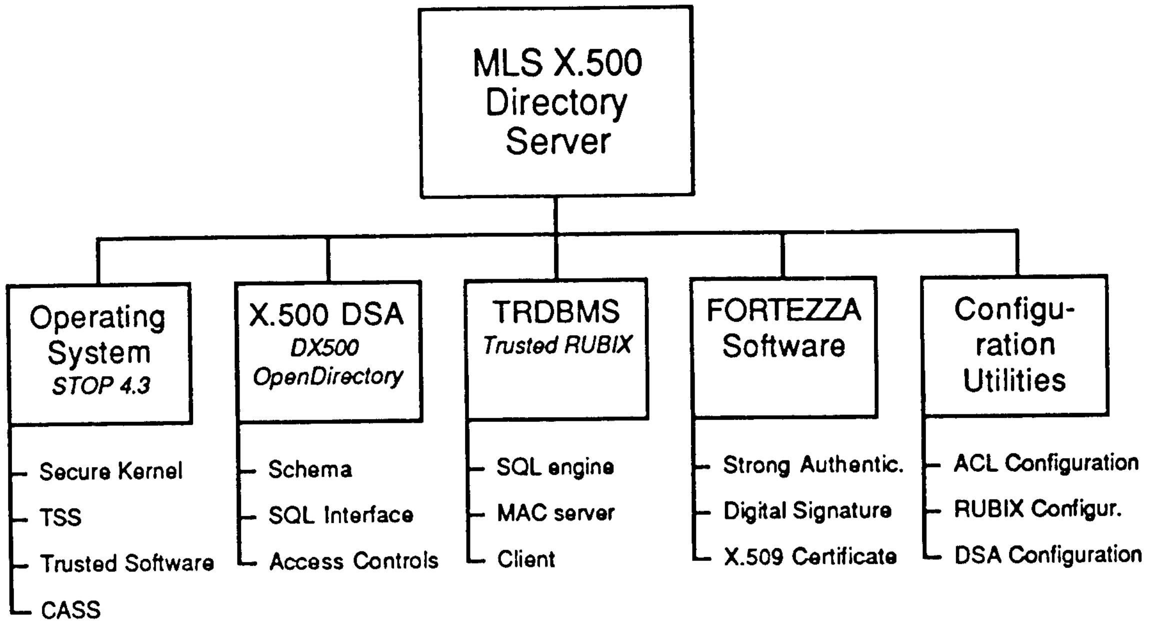


Figure 4-2. Proof-of-Concept CSCIs

These modifications include:

1) Addition of CASS gate software to the standard XTS-300 STOP operating system to enable Ring 3-resident (untrusted) processes to call Ring 2-resident (trusted) processes in a highly controlled manner. This CASS gate has been incorporated into the standard STOP Release 4.3, and is thus not considered a development item.

2) Elimination of specific UNIX System V Release 4.3 dependencies from Trusted RUBIX. The specific changes performed are:

   - elimination of SVR 4.3 dependencies from MAC decision-making code, audit code, and RUBIX RPC code;

   - generic changes to isolation algorithms to eliminate RUBIX capabilities to change levels, as RUBIX-style level changes are prohibited by the STOP OS.

4.2.1 Operating System Architecture

The XTS-300 operating system architecture is built on top of a hardware (Pentium chip) ring mechanism which underpins the security of the operating system by physically isolating portions of system processes from tampering. The XTS-300 implements four isolated rings, or domains. Figure 4-3 depicts the STOP four-ring architecture; the TCB is represented by the lightly-shaded portion. Architectural details, including functions and capabilities, of STOP are described below.

4.2.1.1 Reference Monitor

To enforce the mandatory access policies that make the XTS-300's Secure Trusted Operating Program (STOP) a multilevel secure, multi-compartment/multi-category operating system, the XTS-300 implements the Reference Monitor concept. The Reference Monitor is the mechanism in the operating sytsem that enforces authorized access relationships between the system's subjects (active elements that attempt access, e.g., user processes) and its objects (passive elements such as data segments, processes, devices, which are accessed by subjects).

Subjects in the XTS-300 are user programs in execution, and can be trusted or untrusted. Trust implies the degree of discipline with which a subject was developed and with which it operates. Not all subjects have to be trusted to get the job done. Trusted subjects are used predominantly when it's necessary to manipulate the system's high-integrity databases, or whenever a strictly-controlled circumvention enforced of the system's security rules is performed (as when reclassifying data in a secure guard application); if necessary, they are allowed to perform strictly-controlled overrides of TCB-enforced access control rules. A subject is considered trusted only if its integrity level allows it to manipulate TCB databases, or if it possesses privileges that exempt it from specific TCB access control rules.

The Reference Monitor checks every attempt by a subject to access, or reference, a system object against the Reference Monitor's own list of authorized reference types (which comprises read, write, and execute) which that subject is authorized to perform with regards to the requested object. In this way, the Reference Monitor validates the subject's right to perform the requested type of reference to the requested object (i.e., to ensure that Subject X is indeed allowed to read Object Y).

To ensure the legitimacy of its Reference Monitor, the XTS-300's access validation mechanism is tamper-proof, and is invoked for every reference by a subject to an object. The Reference Monitor is implemented in the system's Security Kernel, which uses the underlying four-ring architecture of the Intel chip to maintain total isolation of various operating system functions from one another, and to isolate operating system code from application code.

4.2.1.2 STOP Components

The STOP operating system comprises two components: the Trusted Computing Base, that enforces security policy, and the Commodity Application System Services (CASS), a UNIX-like application programming environment designed to host unprivileged user applications while providing the TCB features necessary to yield a high level of security and integrity of those applications. The XTS-300 operating system architecture is built on top of a hardware (Pentium chip) ring mechanism which underpins the security of the operating system by physically isolating portions of system processes from tampering. The XTS-300 implements four isolated rings, or domains. In the figure below, the TCB is represented by the shaded portions.

Ring 0, Security Kernel—Most privileged domain, in which resides the Reference Monitor that enforces system security policy. I/O device drivers reside in Ring 0. Small and well-structured to enable complete security evaluation, testing, and verification, the Kernel provides basic operating system services such as resource management, process scheduling, interrupt and trap handling, auditing, and enforcement of mandatory security and discretionary access policies for process and device objects. The security policy is composed of two sets of rules, one governs system security; the other governs system integrity.

Ring 1, Trusted System Services (TSS)— Cannot be called or modified by users. Includes network services, I/O management, file system management, and enforcement of discretionary access policy for file system objects (ie, services not provided by the Security Kernel) provided to both trusted and untrusted system software and applications. The environment provided by the TSS is controlled by the underlying Security Kernel, which enforces mandatory security policy upon the TSS and all other XTS-300 operations.

Ring 2, Trusted Software and Commodity Application System Services (CASS)— Operating system domain; shared between Trusted Software and user-developed trusted processes and the UNIX-like CASS environment. The published interface to Trusted Software is proprietary with UNIX-like features. Ring 2 is the only interface between the application domain (Ring 3) and the underlying trusted domains. Ring 2 can contain trusted and untrusted software; whether a software process is trusted or not depends on its security requirements, ie, whether it has to update a trusted database, and/or whether it has to be exempt from standard STOP access controls. Includes all security relevant functions that operate as independent services (e.g., trusted communications Sockets). In some cases, a Trusted Software function may require the ability to bypass the TCB's mandatory and/or discretionary policy controls. For example, trusted processes enable high-integrity users to set up and modify the file system hierarchy to accommodate the use of high-integrity nodes. Trusted Software functions are available to trusted user processes, and system operators and administrators,for performing security-related system housekeeping such as registering/removing users, assigning passwords, installing and configuring the system, andfor performing other privileged tasks not supported by other STOP components.

CASS is the users' main programming and processing environment. With no privileges to violate security policy, CASS includes an application programmatic interface that provides an implementation of the UNIX System V Interface Definition (SVID), enabling easy UNIX application porting or development on the XTS-300. Only a very few SVID services that violate the NSA-defined security policy have been replaced by a CASS equivalent, or eliminated. In addition, in STOP 4.3, CASS includes a gate (not part of the TCB) that enables Ring 3 processes to spawn Ring 2 processes, a capability required by some commodity MLS applications.

Ring 3, Application Domain—Reserved for untrusted user-developed (or ported) processes. Can run "shrink-wrapped" UNIX applications that comply with the Intel Binary Compatibility Standard (iBCS2).

4.2.1.3 Mandatory Security and Integrity Policies

The subjects in a MLS system are strictly limited to referencing objects according to the NCSC-approved Bell-LaPadula formal mathematical model of computer security policy3. In the XTS-300, this policy is implemented by a set of security rules designed to protect data from unauthorized access. The XTS-300 multilevel TCB implements the Reference Monitor concept and enforces the Bell-LaPadula model, while providing even stricter security * property control. Bell-LaPadula specifies the following mandatory security policy rules:

> *Simple security—A subject may read or execute an object only if the security level of the subject dominates (is greater than or equal to) that of the object.*
>
> *Security * property (read: "Security star property")—A subject may write an object only if the security level of the object dominates that of the subject. The XTS-300 is even more restrictive in its implementation of security * property protection. It allows a subject to write to an object only if subject and object are at the same security level, preventing the problem of a lower-level subject writing higher-level objects that it may not then read or modify.*

Unique in the industry, the XTS-300 TCB also enforces K.J. Biba's integrity policy, a corollary to the Bell-LaPadula security model that enforces the system's mandatory integrity rules. These rules protect information from unauthorized modification (writing), whereas security rules protect information from unauthorized access (reading). As with its security * property enforcement, the XTS-300 provides even stricter integrity * property control than called for by Biba. Specifically, Biba integrity policy enforces the following mandatory integrity rules:

Simple integrity—A subject may read or execute an object (eg, a data file) only if the integrity level of the object dominates that of the subject.

*Integrity * property (read: "Integrity star property")—A subject may write an object only if the integrity level of the subject dominates that of the object (exception: one process may write up to another). The XTS-300 goes a step farther, allowing a subject to write an object only if the integrity level of subject and object match.*

The XTS-300 supports 16 hierarchical security classifications and 64 mutually-independent security compartments or categories, eight (8) hierarchical integrity classifications (four for users, one for operating system domain programs, one for operators, one for administrators, and one to be assigned by security administrator) and 16 mutually-independent integrity compartments or categories. The integrity classifications include at least the following:

```
user < operator < administrator
```

("<" indicates that subject to left is less privileged than subject to right.)

The XTS-300's integrity policy is particularly useful for providing highly protected domains that enable executables to run and configuration files to be read by all processes that need to read them while preventing those objects from being modified (e.g., by malicious logic such as Trojan Horses). The XTS-300 TCB contains privileged programs, such as FSM (File System Manager), that allow users with high integrity privileges to circumvent these rules in a highly controlled and audited manner so that they are able to construct a usable file system hierarchy.

The XTS-300 also enforces a discretionary or need-to-know policy, whereby access to an object is determined by the identity of its subjects and/or the groups to which they belong. The TCB enforces the following discretionary access rule:

Access modes—A subject may access an object in only those mode(s) granted by the owner of the object. Each object shall be assigned permissions (read, write, execute) for the owner of the object, for the members of the owner's group for other specifically identified groups, and for all others.

Each object is referenced by its own unique identifier, and each has its own set of access information and status information. This access information includes the object's subtypes and mandatory and discretionary access attributes, and is the basis upon which the Security Kernel makes its decisions. Specifically, an object's mandatory access information consists of its security level and categories, and its integrity level and categories.

4.2.1.4 Discretionary Access Policy

Object discretionary access information includes:

- object's owner and group identifiers;
- read, write, execute permissions for owner, for members of groups to which owner belongs, and for all other users;

- up to six (6) user and group identifiers and their permissions (read, write, execute);
- object's subtype (subtypes are finer gradations of protection; there may be one or more subtypes per "parent" type).

The TCB follows a set of general rules to determine whether a subject should be granted discretionary access to an object:

- If subject owns object, use specified owner permissions; *if not*
- If entry exists for subject in Access Control List (ACL), use ACL permissions; *if not*
- If subject's current group is the same as group of object's owner(s), use specified group permissions; *if not*
- If there is an entry for group in ACL, use group permissions; *if not*
- If subject has no other specific permissions, use specified "other" ("world") permissions.

4.2.1.5 Philosophy of Protection

All software processes in the XTS-300 are subject to Bell-LaPadula security and Biba integrity rules bounded by the process isolation enforced by the Security Kernel; processes may access information in a ring of the same or lesser privilege, but not in a ring of greater privilege. All portions of the TCB are protected from unauthorized tampering in one of the following ways:

Protection from modification—Security Kernel code and data are protected from modification by any ring other than the Kernel itself. TSS and CASS code and data are protected from modification by processes in any ring other than their own.

Integrity—All TCB program files, databases, and most trusted software processes are protected by setting their integrity level high at operator level or higher. Untrusted users (subjects) are excluded from the TCB by restricting their maximum integrity levels in the user authentication database to less than the integrity levels of TCB objects.

Private segments—Trusted software processes protect their temporary data segments from untrusted software by creating them as private segments. Private segments cannot be shared by other processes. This enhances the system's security by isolating the processes from each other.

Secure path—Before a terminal can communicate with the TCB, the operator must strike the Secure Attention Key to disconnect the terminal from an untrusted process. By allowing only one link at a time between the terminal and any Ring 0, Ring 1, or Ring 2 (trusted or untrusted) process, the XTS-300 completely isolates the trusted communications path from the untrusted communications path. (Terminals may be shared by multiple simultaneous untrusted Ring 3 [application] processes.)

Terminal Subtypes—An unlocked terminal to be used by trusted software is protected from untrusted software by using terminal-unique device subtypes (all under the type "terminal"). When the TCB is entered via the secure path, the secure server removes the terminal's subtype from all untrusted processes associated with the session before it unlocks the terminal. Access to the terminal is restored to untrusted processes only after the operator exits the TCB.

4.2.2 DSA Architecture

The DX500 OpenDirectory™ DSA, a product of Datacraft Technologies Pty Ltd, was selected in large part because of it internal architecture. Datacraft has implemented an X.500 DSA as an integrated relational database management system (RDBMS) appliciation while achieving efficiency and performance through the use of a patented meta-data design and an ANSI SQL interface to an RDBMS (in the proof-of-concept this is Trusted RUBIX). The DX500 OpenDirectory™ DSA provides all of the hierarchical Directory Information Tree processing and object-oriented handling within its application code, while also containing special-purpose database table designs. This design results in high-performance, scaleable X.500 DSA application which inherits the benefits of the underlying RDBMS, including:

- caching
- query optimization
- replication / mirroring
- recovery
- house keeping and tuning utilties

The DSA design is architected to be modular, with clearly identifiable independent processing modules, as illustrated in Figure 4-3.

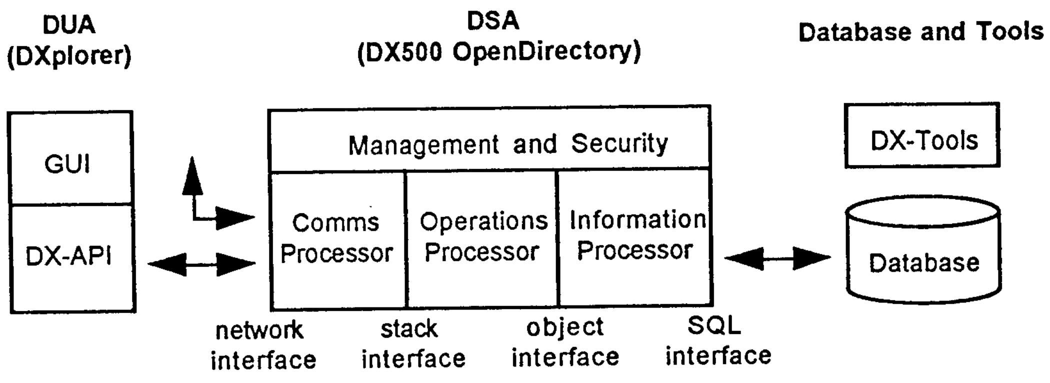


Figure 4-3. DX500 OpenDirectory Architecture

4.2.2.1 DUA Modules

The power of X.500 is visible through the Directory User Agent (DUA). For the end user this means using an advanced Graphical User Interface (GUI). For the application developer, it means having a highly functional, high speed Applications Programming Interface (API). Datacraft's DXplorer™ incorporates both the GUI and DX-API modules. These two modules are described below.

Graphical User Interface (GUI)—presents information to the user in a graphical environment and interacts with the user using intuitive "point and click" mechanisms (described further in the next section). It abstracts much of the X.500 jargon so that information is presented to the user in easy to use forms and hierarchy displays. Internally it uses DXClasses which can be componentised for use in a Delphi environment or converted to applets in the JAVA environment.

DX-API— a portable 'C' high performance API which gives direct access to DAP primitives. The API is suitable for constructing embedded directory clients into office automation products, and other network enabled applications. Underneath the API is a full OSI/RFC1006 communications stack (DAP, ROSE, ACSE, Presentation, Session, Transport). In a Microsoft Windows Environment the API is implemented as a DLL and interfaces to Microsoft's Winsock or compatible TCP/IP stack.

4.2.2.2 DSA Processing Centres

The Directory System Agent (DSA) is highly architected and modular with clearly identifiable and independent processing modules. The methodology uses state-of-the-art "design by responsibility" where independent processing centres are utilised to perform complex tasks. These are briefly described below.

Comms Processor—handles all communications including user protocols (DAP, LDAP), system protocols (DSP, DISP, DOP), management protocols (CMIP, SNMP) and supporting OSI Protocols (ACSE/ROSE, Presentation, Session and Transport). The design is characterised by high performance and efficiency, especially in the areas of protocol coding and decoding and dynamic buffer management.

Operations Processor—co-ordinates the evaluation of X.500 services. It analyses X.500 requests, chooses an execution strategy, checks schema rules, controls sub-operations, assembles results and handles graceful recovery of errors or aborts. It maintains knowledge of all users and service administration limits and can be instructed to intercept or override user controls. It also manages distribution and replication with remote DSA's including, chaining, referrals, knowledge management, shadowing, and results merging.

Information Processor—responsible for data management and the efficient execution of component X.500 services. It is a highly complex module implementing the patented database design and table management strategies. It analyses, requests, resolves aliases, decomposes filters and pre-evaluates expressions, formulates optimal SQL queries, and processes results of queries returned from the RDBMS. It must also cater for BLOBs (Binary Large Objects), minimise memory usage, and avoid any type of inefficient SQL queries (e.g. ordering, aggregates, unions, nulls).

Management and Security—provides general controls, monitoring, and configuration facilities. It supports the processing of authentication controls on users and access controls on data. The Command Line Interface provides the administrative user with a scripting language for administrative tasks, testing and configuration. It also provides a Layer Management Interface for the CMIP and SNMP management protocols.

4.2.2.3 DSA Interfaces

Formal and extensible interfaces separate the major processing modules in a way that maximises encapsulation and minimises coupling. Strict adherence to formal interfaces means that individual processing centres can be broken apart as separate processors, or interfaced to third party products. It also means that developers can work independently thus removing potential project bottlenecks. The major interfaces are briefly described below.

Network Interface—provides external connectivity to the directory using any of the major networking protocols including RFC1006, TCP/IP, UDP, and X.25. This interface could be expanded to use other vendors stacks e.g. IPX, Netbios etc.
The Stack Interface provides a generic asynchronous multi-protocol capability to the DSA and provides functional isolation from the protocol stacks. It allows multiple dissimilar protocol stack handlers, to concurrently access the DSA. It is designed to allow the communications stack to be a separate process if required.

Object Interface—provides a hierarchical object-oriented formal interface which functionally de-couples the back-end database processing. This architecture allows the Information Processor to be a separate process, and provides the possibility for the DX500 to provide the back-end for other vendors products. For example, it is feasible to integrate the Information Processor with a QUIPU (a public domain X.500) directory.

SQL Interface—manages communications with the backend database. ANSI SQL has been used to facilitate portability. It can be recompiled/linked with embedded SQL services to support any SQL database e.g. Ingres, Oracle, Sybase etc. or even network SQL interfaces such as ODBC (Open Database Connectivity).

4.2.2.4 Management

DX-Tools—provide utilities and data management functions such as data preprocessing, data translation and substitution, import/export, dump/reload, update in place and merge/synchronisation. These tools facilitate interworking with external data systems.

Administrative controls—can restrict the number of users bound to the directory, the number of concurrent operations and the maximum size limits and time limits for a user operation.

Engineering traces and diagnostic functions—including a special summary log from which auditing, accounting/billing and statistical information can be obtained.

Alias Integrity—the DSA can enforce alias integrity and can emulate proprietary directory behaviour such as Quipu (the public domain X.500).

Dynamic Schema Configuration—The DSA schema supports dynamic configuration of data types, schema rules and knowledge information. This includes extremely large data types (Binary Large Objects or BLOBs) so that the directory can store multi-media attributes. It can transparently disconnect and reconnect to different databases so that databases can be "hot swapped" while the directory is running. The DSA can even start up without its database, to act as DSP relay e.g. for firewalling applications.

CMIP and SNMP—can be used for remote monitoring.

4.2.2.5 Security

The DSA supports the following security capabilities:

- Authentication
- DAP name and password checking
- DSP and DISP - name, password and address checking
- X.500-1993 standard access controls
- rules based groups

4.2.3 Detailed RDBMS Architecture

Trusted RUBIX consists of three major components, as illustrated in Figure 4-4.

1) Trusted RUBIX kernel
2) SQL engine
3) Untrusted code

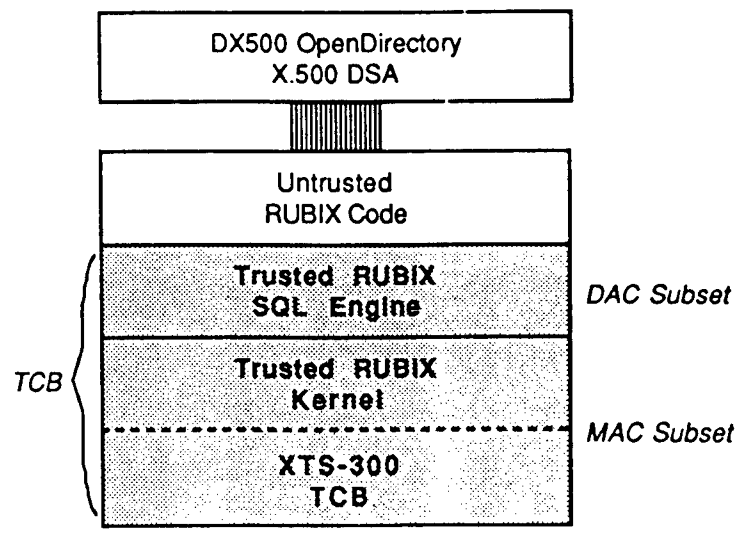


Figure 4-4. Trusted RUBIX Architecture

Trusted RUBIX implements a dual-subset architecture, wherein one subset enforces a mandatory access control (MAC) policy on DBMS objects (i.e., tables, schemata, databases, and tuples), and the other enforces a discretionary access control (DAC) policy.

4.2.3.1 MAC Subset and RUBIX Kernel

The MAC subset consists of the XTS-300 STOP operating system TCB and the Trusted RUBIX kernel, which runs as a trusted subject within STOP. The MAC subset implements the Bell-LaPadula model with databases, relations, and tuples as the system's objects. The DBMS functions implemented in the MAC subset are:

- file management
- buffer management
- relation management
- transaction management
- trusted recovery
- logging
- auditing of MAC operations.

The architecture of the portion of the MAC subset that comprises the Trusted RUBIX kernel but not the underlying TCB and hardware (hereafter referred to as *RUBIX kernel*) is based on the concept of a *protected subsystem* in which all MAC-protected data are stored in one or more volumes, which are single-level operating system objects. To support fine-grained multilevel objects such as tuples, labels are attached to individual database elements within each operating system object.

These labels are DBMS labels, not operating system labels; the operating system views these DBMS labels strictly as data, and attaches no security significance to them. The DBMS is trusted to properly associate and maintain the label of each item, and to correctly interpret the labels so that, in cooperation with the operating system kernel, the security policy is correctly enforced.

The RUBIX kernel is implemented as a separate operating system process. The resources protected by the RUBIX kernel are protected from external access by setting them to a reserved "user" security level. When created, processes in the RUBIX kernel will be set to this reserved "user" security level via the STOP *load_process* system call. In addition, processes at this RUBIX-specific "user" security level will be granted privileges to communicate with untrusted processes via pseudo-ttys.

A notable characteristic of the RUBIX-specific "user" security level is that it includes a unique category reserved for subjects and objects in the RUBIX kernel. Processes outside the RUBIX kernel cannot have this reserved category in their labels, and thus are prevented from directly accessing RUBIX kernel subjects and objects.

In these ways, Trusted RUBIX uses the underlying STOP mandatory access control mechanisms to isolate RUBIX DAC subset subjects and objects from STOP subjects and objects.

The executables that make up the RUBIX kernel are MAC-protected from unathorized modification by setting their mandatory security and integrity levels via STOP's *tp_edit* utility. In addition, all RUBIX kernel programs are installed with the appropriate stop DAC permissions to prevent unauthorized access.

Since the RUBIX kernel is implemented as a separate process, it uses an interprocess communication (IPC) interface. In the STOP implementation of Trusted RUBIX, this IPC interface is provided by STOP's IPC mechanism and pseudo-tty mechanism. Conceptually, this IPC interface consists of a set of procedures. From the client's point of view, all it must do to access the services of the RUBIX kernel is to link to the library containing these procedures. These procedures are implemented as a form of remote procedure call, with two versions of each procedure: (1) client-side procedure that runs in the DAC subset domain, and (2) server-side procedure that runs in the MAC subset domain.

When an initialization routine is called, the *rxkernel* server process is invoked, and a pseudo-tty is established between the client and server processes for synchronous communication. When a program calls one of the client-side access functions, the call's arguments are collected and passed, along with a procedure identifier, to the server side. The client procedure then blocks and waits for a response. On the server side, a dispatcher function examines the procedure identifier and calls the appropriate server-side procedure with the communicated arguments. The server-side procedure validates the arguments, then performs the called function. When this call returns, the dispatcher sends back any return values to the client procedures. The client procedure then returns like a normal procedure call. The dispatcher then blocks, awaiting the next request. The communication channel is brought down by a termination routine.

4.2.3.2 DAC Subset

The DAC subset consists of the Trusted RUBIX SQL engine. This subset depends on the MAC subset, and implements a discretionary access policy that enforces further access restrictions above and beyond the mandatory access policy enforced by the MAC subset. The DAC subset implements a DAC policy on databases, schemata, relations, views, indexes, and columns. The DBMS functions implemented in the DAC subset are:

- query parsing
- query optimization
- execution of query plans
- join algorithms, sort algorithms, group-by, etc.
- integrity constraints
- view management
- index management
- audit of DAC objects.

The architecture of the DAC subset is also based on the concept of a protected subsystem. As with the RUBIX kernel, the DAC subset is implemented as a separate process. The DAC subset protects its resources by using the underlying trusted STOP DAC mechanism. All DAC subset resources including database volumes are protected from external access by making them accessible only to subjects in a reserved operating system group called *"rubixTP"*. Once these protections are in place, the underlying operating system will not allow access to the resources unless the effective group-ID of the accessing process is *rubixTP*. Upon invocation of the DAC subset, the mechanism in STOP for setting group IDs is used to set the effective group-ID of the DAC subset process to *rubixTP*. This allows the process to access protected resources.

The DAC subset uses the same mechanism as the RUBIX kernel to protect its executables from tampering with a similar IPC interface to that used in the MAC subset.

4.2.3.3 Reference Monitor

Each RUBIX TCB subset must satisfy three reference monitor requirements:

1) The reference monitor cannot be bypassed. Subjects external to the subset cannot access protected resources without having that access mediated by the subset. The RUBIX kernel prevents external subjects from accessing its data by labelling those data at the RUBIX-specific "user" level. The DAC subset prevents external subjects from accessing its data by storing them so that only members of the reserved *rubixTP* group can access them, and restricting membership to the group to processes within the DAC subset.

2) The reference monitor must be tamper-resistant. Bot the RUBIX kernel and the DAC subset use the same mechanisms to protect themselves. Since RUBIX kernel subjects and DAC subjects run as separate processes, they are protected from tampering by the underlying STOP process isolation mechanisms. RUBIX kernel processes have an extra measure of protection because they run at the RUBIX-specific "user" level, and are therefore further isolated from untrusted processes. However, because of security policy restrictions in the underlying STOP TCB, RUBIX within STOP may have to be implemented somewhat differently to enable interprocess communications within RUBIX.

3) The reference monitor must be small enough to be subject to analysis and tests the completeness of which can be assured. The Trusted RUBIX MAC and DAC subsets are designed to be modular and small enough so that correctness can be established.

4.2.3.4 RUBIX Untrusted Processes

Outside the STOP TCB (i.e., in Ring 3 rather than Ring 2) will reside the untrusted RUBIX components. These include:

- i-sql
- dynamic sql
- embedded-sql
- cli
- client-side rda.

4.2.4 FORTEZZA I&A Design

FORTEZZA PCMCIA-card based encryption technology and MISSI FORTEZZA software are being used to implement strong authentication in the DX500 OpenDirectory DSA. The FORTEZZA software (version 3.0.1), and a Spyrus PCMCIA card reader and Cryptographic Interface (CI) library (version 1.52) have been installed on the XTS-300. J.G. Van Dyke and Associates is developing a library of interface software to provide DSA access to the appropriate authentication library functions. These, in turn, will access information contained on the FORTEZZA card via the CI library.

4.3 Integrated Software Architecture

4.3.1 Directory Server Internal (Component-to-Component) Interfaces

Figure 4.5 depicts how the different CSCIs will be implemented within the STOP operating system's four-ring architecture.

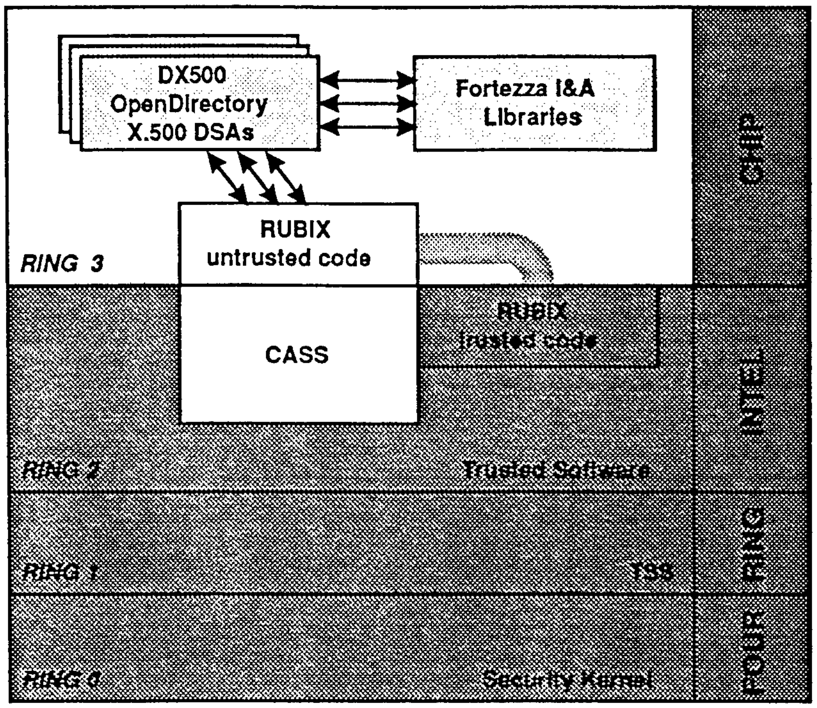


Figure 4.5. Mapping of CSCIs into STOP OS

4.3.1.1 Trusted RUBIX Interfaces

Calls to the following DX500 OpenDirectory file system directories and their contents must be modified to replace the standard Ingres RDBMS interfaces with Trusted RUBIX interfaces:

As delivered	*Change to*
datacraft/dsa/dip/ingres *tAlias.sc, tAttr.sc, tBlob.sc, tDit.sc, tEntry.sc, tInfo.sc, tName.sc, tSearch.sc, tTne.sc, tUtils.sc*	datacraft/dsa/dip/rubix *tbd*
datacraft/dsa/utils/ingres *setupDB.sc*	datacraft/dsa/utils/rubix *tbd*
datacraft/dsa/utils/tools *dxnewdb, dxshadow, dxtunedb, dxupdate, dxreload*	RUBIX command line interface (now uses Ingres command line interface) *tbd*

In addition, though DX500 OpenDirectory uses ANSI standard data types, i.e. no proprietary Ingres types, the DSA performs "sounds-like" searches using SQL pattern matching. Part of the integration effort will involve replacing DSA calls to Ingres' "sounds-like" capability to Trusted RUBIX's ANSI/XOpen standard "LIKE" operator.

4.3.1.2 DSA and Trusted RUBIX Security Requirements

The processes in the MLS Directory Server running on the XTS-300 STOP operating system require the assignment of security and integrity levels, user IDs, group IDs, and permission sets. The following diagram shows the recommended assignments of these security attributes to the MLS Directory Server components.

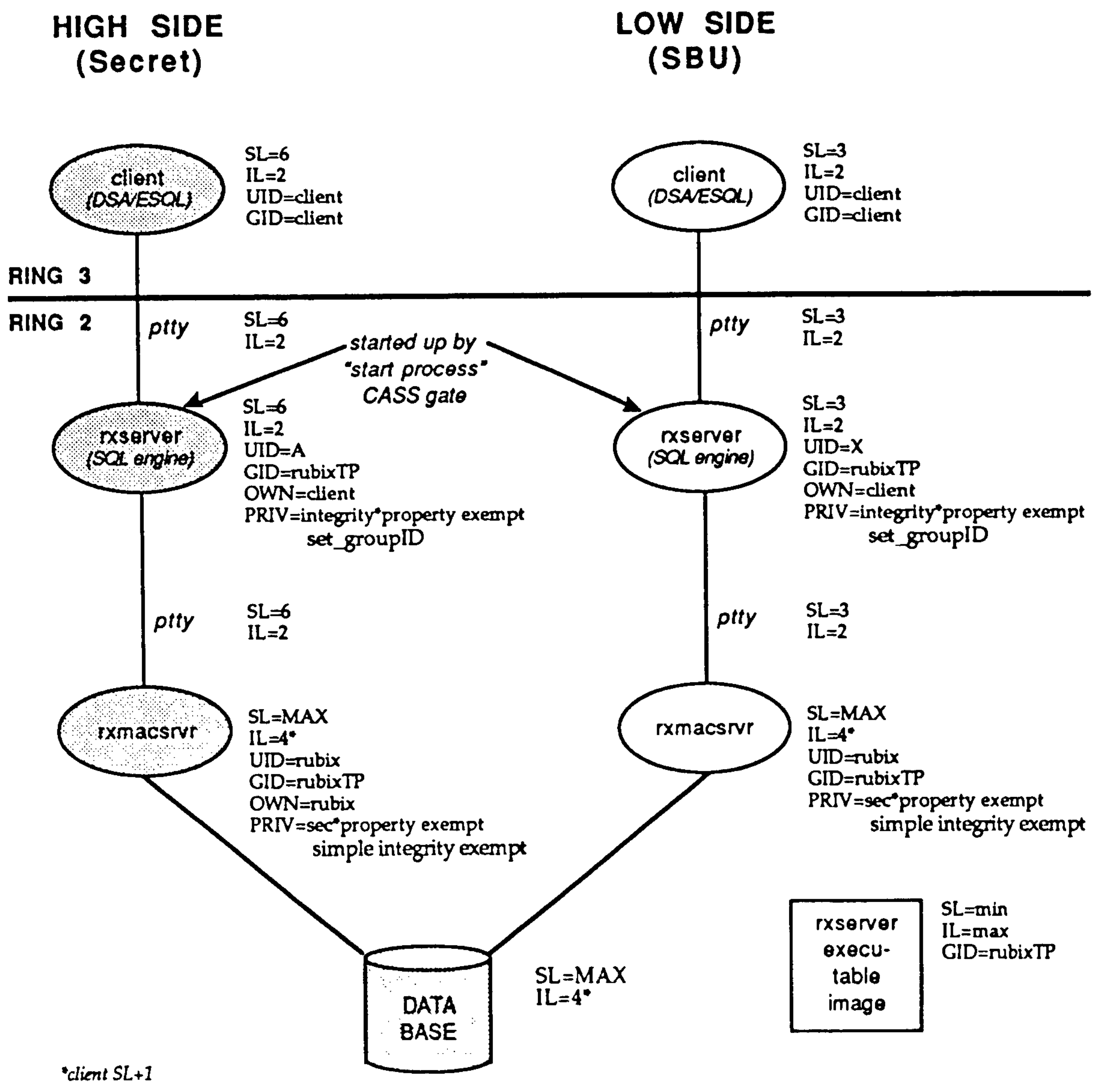

HIGH SIDE (Secret)
LOW SIDE (SBU)
client (DSA/ESQL)
SL=6 IL=2 UID=client GID=client
client (DSA/ESQL)
SL=3 IL=2 UID=client GID=client
RING 3
RING 2
ptty
SL=6 IL=2
started up by "start process" CASS gate
ptty
SL=3 IL=2
rxserver (SQL engine)
SL=6 IL=2 UID=A GID=rubixTP OWN=client PRIV=integrity*property exempt set_groupID
rxserver (SQL engine)
SL=3 IL=2 UID=X GID=rubixTP OWN=client PRIV=integrity*property exempt set_groupID
ptty
SL=6 IL=2
ptty
SL=3 IL=2
rxmacsrvr
SL=MAX IL=4* UID=rubix GID=rubixTP OWN=rubix PRIV=sec*property exempt simple integrity exempt
rxmacsrvr
SL=MAX IL=4* UID=rubix GID=rubixTP PRIV=sec*property exempt simple integrity exempt
DATA BASE
SL=MAX IL=4*
rxserver executable image
SL=min IL=max GID=rubixTP
*client SL+1

Figure 4.6. Security Posture of DSA and Trusted RUBIX on XTS-300

DEFINITIONS:
SL = security level
IL = integrity level
UID = user ID
GID = group ID
OWN = owner (owner SL=process SL)
PRIV = privileges required by process
ptty = pseudo-tty

Note that in the diagramme, the *client* process actually includes all processes of DX500 OpenDirectory DSA, the FORTEZZA strong authentication processes, and the Trusted RUBIX ESQL interface. From the Trusted RUBIX standpoint, this collection of processes form a single monolithic *client* process.

For the MLS Directory Server, security levels of the various processes are assigned arbitrarily based on a two-classification level model. In other applications—or in an MLS Directory Server with more than two classification levels, the true model is one of system high being considered at MAX security, and system low being considered at MIN security, with a range of hierarchical security levels between MIN and MAX assigned to classification levels between system high and system low. The same holds true of the hierarchical integrity levels, with the understanding that the highest integrity level available in Ring 3 is il3. In some RUBIX applications, the integrity of the client processes should be set at different levels to ensure the inability of those processes to interfere with each other. This is not considered a risk in the MLS Directory server, so the two client processes are both set at il2.

4.3.1.3 DSA-FORTEZZA Interfaces

The standard DX500 OpenDirectory I&A process will be modified to use the J.G. Van Dyke & Associates FORTEZZA-based strong authentication libraries to provide strong authentication processing on binds between the MLS Directory Server and external DUAs and DSAs. The Van Dyke FORTEZZA I&A software will provide the interface between the Cryptographic Interface (CI) library on the XTS-300 and the DSA. Modifications to the DX500 OpenDirectory's standard authentication process will be implemented.

When the DSA is initialized, it will call *msp_login* to log into the FORTEZZA card and retrieve information essential to perform validations. This log-in session will be maintained for the duration of the DSA session.

DAP bind authentication: During the Directory Access Protocol bind process, the DSA will recognize the request for strong authentication and call the FORTEZZA I&A interface software. This I&A software will validate the credentials contained in the DAP bind argument. The interface software will ASN.1-decode the requestor's credentials, then call *msp_val_objects* to perform the actual validation. Access to the DSA's services will be denied if the requestor's credentials cannot be validated.

DSP bind authentication: During the Directory System Protocol (DSP) bind process for chaining between DSAs, the DX500 OpenDirectory DSA will request strong authentication and call the I&A interface software to prepare the credentials for the DSP bind argument. The interface software will ASN.1-encode the DSA's credentials, then call *msp_generic_sign* to generate the necessary digital signature. When the DSA receives the ID and credentials of the peer (external) DSA, it will ASN.1 decode those credentials and call *msp_val_object* to perform the actual validation.

Figure 4-7 illustrates the use of FORTEZZA-based strong authentication by a DSA within the MLS Directory Server.

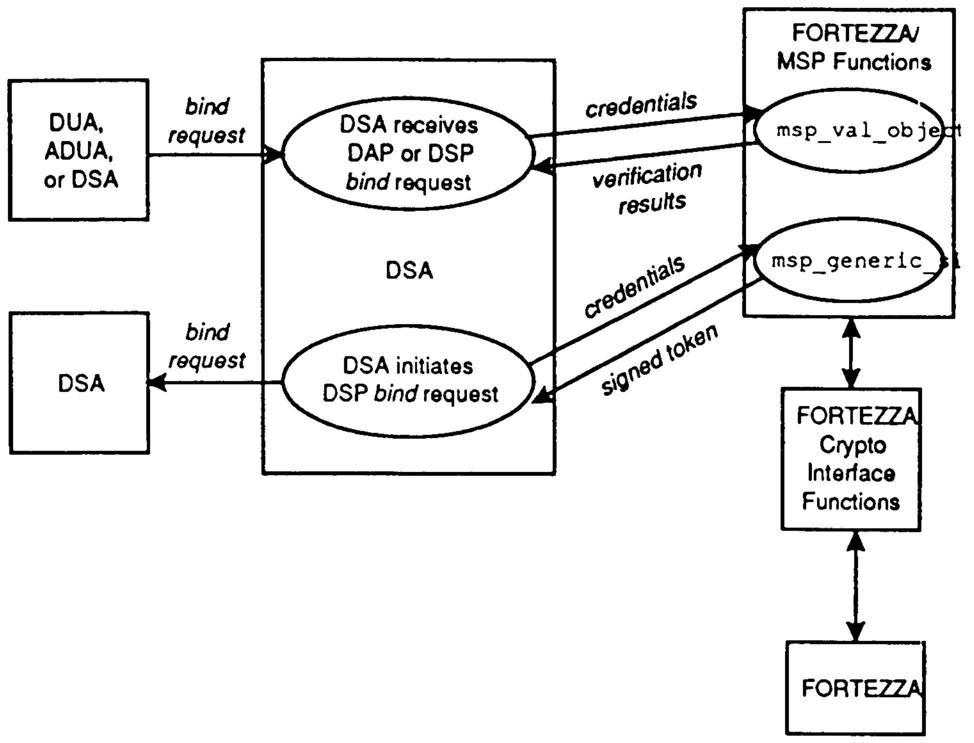


Figure 4.7. DSA Integration with FORTEZZA-based Strong Authentication

4.3.2 Directory Server External Interfaces

As noted, by using the standard DX500 OpenDirectory X.500 communications protocols (DAP, DSP), the MLS directory server will be able to communicate to external X.500 DSAs and DUAs that also support those protocols in a standard manner (i.e., X.500-1993 compliant). Within the MLS directory server, the "external" interface between the DSAs and the Trusted RUBIX RDBMS will be ANSI SQL.

4.3.3 Directory Server Data Stores

The data store to be used by the MLS directory server is the Trusted RUBIX database, which will be populated with directory information through a single-level Directory User Agent (DUA), via the corresponding level of DX500 OpenDirectory DSA. In this way, the RUBIX database becomes the DSA's Directory Information Base (DIB). During integration, attention will be paid to the implications of having multiple DSAs use the same RUBIX database as if they "owned" it, or more accurately, owned the portions of the database to which they are authorized access (e.g., based on classification level).

The DSA is highly optimized to use RDBMS primary indicies, enabling it to perform very fast on any RDBMS. Any additional tuning is provided by the RDBMS itself, in its ability to rebalance its indicies, e.g., B-trees, after mass update. Also important is the ability of the RDBMS's query optimizer to understand the distribution of data within its tables.

4.3.4 Directory Server MLS Schema Design

No schema design at the RDBMS level is required, per se. Instead, the RDBMS schema is generated by a DSA schema translation utility, which initializes new DIT/DIBS by performing SQL CREATE TABLE commands. Regardless of the X.500 schema contents, the SQL TABLE structure will be the same, as the DSA is actually storing meta-data (this is an aspect of the DX500 OpenDirectory patented design). Thus the schema design issues occur at the X.500 DIB level, not at the RDBMS level.

For the proof of concept, the schema design will be kept simple for demo purposes. We will not attempt to resolve optimal schema design issues in this project, but will recommend that these issues be resolved in the proposed follow-on work (see Section 4.6). The draft directory schema design for this project is attached as Appendix A of this document. This draft schema design is subject to change. In any case, the proof-of-concept schema will be designed to support DMS data elements, though the full DMS schema will not be implemented at this time.

4.4 CONCEPT OF OPERATION

4.4.1 Inbound Data Flow

4.4.1.1 Lookup and Update Requests

Lookup or update requests received by the MLS Directory Server from an external DUA will be handled at the level of the external DUA.

4.4.1.2 Chaining of Lookups and Updates

Lookup or update requests chained to the MLS Directory Server from an external DSA will be handled at the level of the external DSA.

4.4.1.3 Shadowing from Other DSAs

The proof-of-concept MLS Directory Server will support DISP for replication using shadowing. For shadowing *to* the MLS Directory Server, the MLS Directory Server will have to authenticate the security level of the external DSA that asks to shadow information to the MLS Directory Server. Based on this authenticated security level, the MLS Directory Server will consider the shadowed information as existing at the authenticated security level of the source DSA; for example, all data shadowed from a Secret DSA will be stored at Secret in the MLS Directory Server's multilevel DIB.

4.4.2 Outbound Data Flow

4.4.2.1 Responses to Lookup and Update Requests

Responses to lookup or update requests will be returned to the requesting DUA/DSA at the level of that DUA/DSA. This will involve the implicit upgrading of data stored in the DIB at levels lower than that of the requesting DUA/DSA.

4.4.2.2 Chaining of Lookups and Updates

Lookup or update requests that cannot be satisfied from its own (local) DIB by the MLS Directory Server will be chained only to an external DSA(s) at the same level as the external DUA or DSA that originated the request.

4.4.2.3 Shadowing to other DSAs

Directory information may be shadowed by the MLS Directory Server to external DSAs in one of two ways:

1) Each portion (level) of information will be shadowed to an external DSA at that level; the entire DIB will not be shadowed.

2) Each portion (level) of information will be shadowed to an external DSA whose level dominates that of the directory information being shadowed; in this way the entire DIB may or may not be shadowed, depending on the highest level of information in the DIB and the level of the external DSA to which the DIB will be replicated.

See Section 4.6 for further discussion of multilevel shadowing from the MLS Directory Server.

4.4.3 DIB Storage and Retrieval

4.4.3.1 Storage of Directory Information

In response to an update request (expressed in DAP or DSP), new (or modified) data will be stored in the MLS Directory Server's local DIB at the level of the DUA or DSA requesting the update. As enforced by the STOP operating system's Security Policy, the internal DSA can bind only to external DUAs/DSAs at its own security level. Thus all update requests handled by a particular internal DSA will be at the level of that DSA. The Trusted RUBIX database manager will label the data entry based on the security level of the internal DSA from which the data is received. Communication between the internal DSA and the RDBMS will be expressed in ANSI SQL. The SQL data will be stored with a classification label reflecting the level of the internal DSA (and, thus, the data), and all future handling of that data will be predicated on the restrictions imposed by that classification label.

4.4.3.2 Retrieval of Directory Information

In response to a lookup request (expressed in DAP or DSP), data may be retrieved from the MLS Directory Server's local DIB by any internal DSA at a level that dominates the level of the data. As enforced by the STOP operating system's Security Policy, the internal DSA can bind only to external DUAs/DSAs at its own security level. Thus the data retrieved to satisfy any lookup requests handled by a particular internal DSA must be dominated by the level of that DSA. Communication between the internal DSA and the RDBMS will be expressed in ANSI SQL. The Trusted RUBIX database manager will make its security policy decisions to release or deny release of the requested SQL data based on the label it applied to the data when the data were stored.

4.5 Limitations of Proof-of-Concept

The MLS Directory Server proof-of-concept project is not intended to produce an operational-ready MLS Directory Server, but to perform all of the integration of components necessary to form the basis for modification/enhancement to create an operational-ready system. Several limitations of this system are imposed by the incomplete nature of International Standards Organization (ISO) X.500 security standards (e.g., Standard Operational Security Enhancements, currently in draft form) and ACP-133.

Other limitations are created by the draft nature of MISSI Trusted X.500 Directory Server requirements and the embryonic nature of the DMS operation, particularly as pertains to the actual physical operation of multiple security enclaves, i.e., will these be on physically separate system high networks, or on a single physical network with logical system high separation based on X.509 and FORTEZZA encryption separation? It makes little sense to attempt to anticipate the DISA solution to this problem, or to anticipate the final versions of what are currently draft standards. Thus, we have prioritized our proposed future enhancements so that those which can be made confident that no changes to X.500/ACP-133 security standards, MISSI Trusted Directory Server Requirements, or DMS operations will make such enhancements invalid or obsolete.

Due to these considerations, and some limitations of the hardware/software components of the MLS Directory Server in their current form, the proof-of-concept MLS Directory Server has the following limitations that must be overcome before it can be considered ready for operational deployment:

4.5.1 Classification Labels Derived from Physical, not Logical, Connections

Security labels are derived from the level of the network connection between the external X.500 object and the XTS-300. There is no logical labelling mechanism at this stage, because it is unclear how DMS intends to ultimately label and separate its enclaves—i.e., logically or physically.

4.5.2 No Support for Multilevel Chaining of DAP and DSP Requests

Multilevel chaining, whereby requests that cannot be satisfied by the MLS Directory Server are chained to multiple directory servers at all levels dominated by the original request, is not yet supported. This will require the development of a privileged process to reclassify and replicate the original request so it can be distributed to multiple external DSAs, and also a process that will collect and aggregate responses from the multiple external DSAs into a single system-high response to the original requestor.

4.5.3 No Approach for Shadowing to Single-Level DSAs

The proof-of-concept MLS Directory Server will support DISP for replication using shadowing. For shadowing *to* the MLS Directory Server, the MLS Directory Server would have to authenticate the security level of the external DSA that asks to shadow information to the MLS Directory Server. Based on this authenticated security level, the MLS Directory Server will consider the shadowed information as existing at the authenticated security level of the source DSA; for example, all data shadowed from a Secret DSA will be stored at Secret in the MLS Directory Server's multilevel DIB.

It is unclear exactly how shadowing (replication) of information to single-level DSAs from the MLS Directory Server would be accomplished. This is a security and operational policy issue rather than a technical restriction, and until an approach is defined, it is difficult to implement such a shadowing capability in the MLS Directory Server in a way that will satisfy all potential users.

4.5.4 FORTEZZA Implementation Limitations

Limitations in the MISSI FORTEZZA architecture and in the XTS-300 hardware architecture restrict us currently to implementing a single dual-card PCMCIA FORTEZZA reader. In future releases, we hope to support two such readers, but this will still limit the number of security levels supportable by the MLS Directory Server to four (4)—one per card. Also, the MISSI FORTEZZA architecture represents a potential performance bottleneck. It is hoped that the MISSI program will devise a more performant, less limiting solution, such as a FORTEZZA card "bank" or a software-based FORTEZZA solution that can be used in high-assurance systems. Until such restrictions are resolved, the number of different levels of data that can be stored in the MLS Directory Server will be limited by the number of FORTEZZA cards it can accommodate.

4.5.5 No Support for not-yet-Finalized X.500 Standard Security Enhancements

ACP-133 and X.500 Standard Security Enhancements have not been implemented, as both standards are still in draft form. Many features under consideration for ACP-133 are being targeted at the X.500-1997 standard. It is our intention to obtain these enhancements from Datacraft in future releases of their standard DX500 OpenDirectory product, and not to have to implement them ourselves, if possible.

4.5.6 Strong Authentication Limited to Binds; no Authentication of Operations

Strong authentication is currently the only security policy checking performed to validate the permissions of the external DSA or DUA. It is presumed that a DSA or DUA who provide the necessary credentials to allow them to bind to the MLS Directory Server's internal DSA should be permitted to perform whatever directory operations they attempt—the only restriction being that the MLS Directory Server will enforce the separation of permissions between DUAs and ADUAs (i.e., authenticated DUAs will only be allowed to perform lookups; authenticated ADUAs will be allowed to perform lookups and updates). Beyond this, no security policy filtering of individual DSP and DAP requests is performed, nor is an external directory system's permission to perform any specific directory operation validated (beyond the DUA "lookup only" restriction). This kind of security policy filtering is expected to be developed by the MISSI program for the DMS Guard and Secure Network Server. When these security policy filters are developed, we propose to implement them in the MLS Directory Server to increase its utility and security policy enforcement capabilities, in essence having it serve "double duty" as a DSA and an X.500 Guard.

4.5.7 Limited Directory Schema Implementation

There are three basic approaches to the design of an MLS directory schema, each with its own potential problems. The simplest approach is to create two separate schema trees, one Sensitive-but-Unclassified and the other Secret. This is how the schema in the Proof-of-Concept is implemented, but is impractical for an operational directory as it will doubtless entail much replication of data in both trees, with the only differences between trees caused by data unique to the particular tree's classification level.

We propose to study the schema design alternatives, including the security and performance challenges posed by each, and redesign the MLS Directory Server DIB schema in accordance with the best choice. However, even resolving these issues will not solve the problem completely, as the current DMS X.500 Directory Baseline Schema, 23 February 1996, presumes a schema at a single classification level, rather than a schema that will accommodate directory information at multiple classification levels. Until the DMS program designs a directory schema for MLS DIBs, any schema redesign work in our MLS Directory Server will necessarily be guess-work in anticipation of an updated or alternative MLS DMS Directory Schema.

4.6 Proposed Enhancements to Proof-of-Concept

We propose to begin implementing the following short-term enhancements immediately upon completion of the proof-of-concept development efforts. Implementation of these enhancements will provide the functionality required for the MLS Directory Server "operational-ready", i.e., ready to function in an operational environment such as the DMS, and ready to be accredited for operation in such an environment. Short-term enhancements will not include those which rely on the finalization of international or U.S. X.500 standards, or the publication of U.S. MLS Directory Server Requirements. The enhancements to satisfy these standards and requirements are addressed in Section 4.5.3, "Longer-Term Enhancements". It is possible that standards/requirements may be finalized soon enough for their implementation in the MLS Directory Server to be completed ***before*** some of the short-term enhancements. What is meant, then, by "short term" and "longer term" in this context is the relative start date for the enhancement, not the completion date.

4.6.1 Proposed Short-Term Enhancements

4.6.1.1 Implement Multilevel Chaining to External DSAs

In the proof-of-concept, chaining is supported only at the security level of the DUA request which the MLS Directory Server must chain to another DSA to fulfill. This means that to fulfill a SECRET DUA's lookup request, the MLS Directory Server will only chain to a SECRET external DSA, even if the information required by the requesting SECRET DUA actually exists on a lower-level DSA.

To support chaining from the MLS Directory Server to external DSAs operating at multiple security levels, we propose to integrate and/or develop a privileged process that will enable the MLS Directory Server to chain a request onward to multiple DSAs at each security level dominated by the originator of the request (DSA or DUA). Thus, to fulfill to a SECRET DUA request, the MLS Directory Server would not only chain to an external DSA at the SECRET level, but also to additional external DSAs at the Confidential and Sensitive Unclassified levels, and at the Unclassified level, if security policy permits.

In the first phase, we will address the chaining of *lookup* requests. In the second phase, we will consider the security implications of chaining *update* requests at multiple levels, as allowing such multilevel update chaining would run contrary to the "write only at same level" policy enforced by the MLS Directory Server itself. If it is determined that supporting multilevel update chaining is acceptable, we will extend the capability of the privileged process developed for multilevel lookup chaining to support it.

4.6.1.2 Analyze and Implement Options for Two-Way DSA Shadowing Capability

We propose to analyze the options for supporting two-way DSA shadowing from the MLS Directory Server to single-level DSAs, and to document the most logical shadowing security policies. For example, the MLS Directory Server could limit its shadowing of data to data at the same security level as the subscribing single-level DSA; for example, a subscribing SBU DSA would receive only data that exists at the SBU level in the MLS Directory Server. This would be approaching shadowing as an *update*, i.e., enforcing the MLS Directory Server's security policy of updating only at the same level as the external DSA (to which the data are being shadowed). Alternatively, shadowing operations could be treated in the same way as *lookup* operations, with the MLS Directory Server satisfying a given shadowing request by shadowing information stored in its MLS DIB at all levels dominated by the external DSA. In this way, an SBU DSA would receive (via DISP) only data stored in the MLS DIB at SBU, but a Secret DSA would receive data stored at SBU, Restricted, Confidential, and Secret, with the lower levels of data essentially upgraded to Secret by the same Trusted RUBIX implicit upgrade process used to satisfy lookup requests from higher-classified DSAs and DUAs.

Once the logical options are defined, we then propose to implement whatever trusted processes or configuration controls are necessary within the MLS Directory Server to enable it to be configured to support each of the recommended options, allowing user organizations to take the final decision as to which option best suits their operational environment.

We also propose to resolve the issue of shadowing between the MLS Directory Server and other MLS DSAs. In this scenario, multiple associations may be required to distribute information from the MLS Directory Server and the subscribing MLS DSA at only the classification levels supported by the subscribing DSA

While inbound shadowing to the MLS Directory Server is not considered a problem, as data shadowed from an external single-level DSA would be considered to all be at the same level (the level of the DSA), and would be stored in the MLS Directory Server at that level, this does not address the issue of *actual* security levels of data stored in system-high directories; the MLS Directory Server can only be expected to work with trusted labels (see x.3.2.7) to determine the classification of incoming data. In the current system-high enclave DMS environment, the MLS Directory Server cannot to expected to be able to distinguish the actual levels of incoming data stored in system-high systems; it can only discern the level of the system-high network from which those data are received.

4.6.1.3 Prototype a MLS DMS Schema Design

The current DMS Schema does not currently address multilevel objects, but presumes objects all at one security level. For the MLS Directory Server to implement the DMS Schema, security labels will have to be assigned to objects and attributes in the current single-level Schema. We propose to determine the most likely security labels for various types of data in the DMS schema, and to undertake a prototype Schema object and attribute labelling effort, to provide a working prototype of an MLS DMS Schema. The other part of this prototype effort will be to analyze several alternatives to designing an MLS Directory Schema. In our analysis, we will prototype at least the following: (1) single complex directory tree with both low and high branches (Figure 4-8) and (2) a single system-low tree with multiple leveled values of attributes stored at various places on the tree (Figure 4-9). These prototypes will help us determine which approach is the least problematical, understanding that both approaches present problems in moving up and down the tree, from low to high, which, depending on the operation taking place (read or write), will require violation of the Bell-LaPadula security. We propose to resolve such security issues in designing and prototyping the optimal MLS DMS schema for the MLS Directory Server.

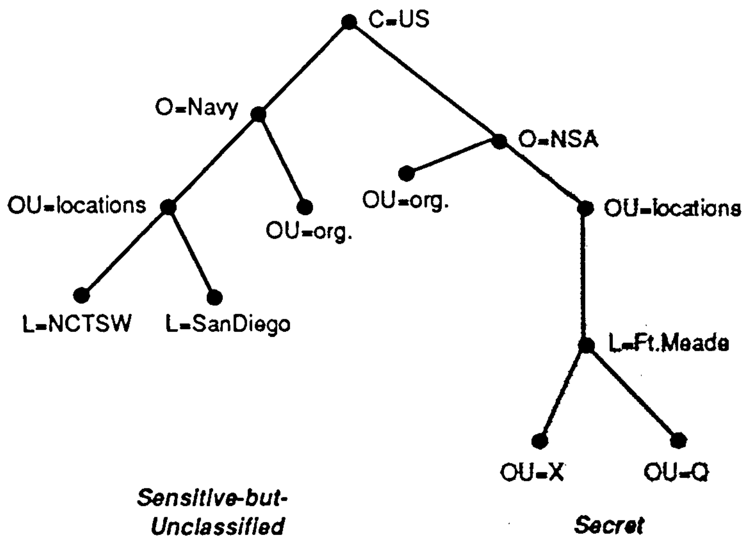


Figure 4.8. Multilevel Directory Tree

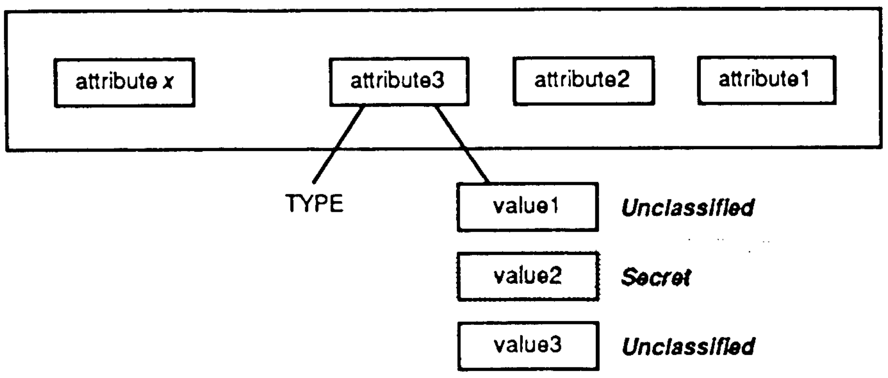


Figure 4.9. Schema with Multivalued Attributes

The potential problem with both of these approaches is in moving up and down the tree, from SBU to Secret, which, depending on the operation taking place (read or write), will be in violation of the Bell-LaPadula security model enforced by the RDBMS. We propose to resolve such security issues in designing and optimal schema for the MLS Directory Server that will satisfy DMS and other MLS directory schema requirements.

4.6.1.4 *Prototype Security Labels Based on Logical Connections*

For the proof-of-concept, each physical network—and all systems on that network—connected to the XTS-300 is presumed to be operating at single security classification level. The security label to be recognized by the MLS X.500 Directory Server for each external DUA and DSA on a particular physical network will be derived from the level of that physical network. This limitation is imposed by the current DMS architecture, with its system-high enclaves and lack of security labelling of data.

We propose to prototype a mechanism for applying logical security labels to the X.500 data flowing over a physical connection, possibly using ACP-133 combined with X.509 certificates and FORTEZZA encryption support to create a trustworthy mechanism for security labelling of X.500 data. This prototype mechanism will be used to strongly authenticate and track the security level of the logical connection for the duration of the association between the MLS Directory Server and the external DUA or DSA.

This kind of logical security labelling will enable the MLS X.500 Directory Server to operate environments such as that envisioned for the future DMS, wherein multiple security levels of data are transmitted over the same SBU physical network, with separation by cryptographic (and possibly other) means.

After prototyping a mechanism for logical labelling of X.500 data, we propose to implement a privileged process in the MLS Directory Server for reading the logical label information and taking security policy decisions for distribution/handling of the multilevel data within the Directory Server based on the logical label, instead of the physical network label which the logical label will override. We will also develop a means of preserving the logical label on both inbound and outbound DSP, DOP, and DISP communications for the duration of the chaining or shadowing association (bind) between the MLS Directory Server and external DUAs/DSAs.

4.6.1.5 Implement MISSI Protocol Filtering, Dirty Word Scans, etc.

As Wang, under contract to the NSA MISSI program, implement the DAP, DSP, DISP, and DOP protocol filters for the DMS X.500 Guard (over the next 12 months), we propose to implement the same filters within the MLS Directory Server to ensure that any requests travelling between DMS enclaves via the MLS Directory Server have been verified to conform to MISSI standards for such interenclave X.500 exchanges. Once these filters are implemented, the MLS Directory Server could provide an alternative to the Secure Network Server or DMS Guard. We feel that the current DMS/MISSI architecture's multiplicity of single-level enclaves served by single-level DSAs and interconnected via X.500 Guards will ultimately lead to an overcomplexity of directory, security, and network management requirements. By incorporating guard capabilities into the MLS Directory Server, we can create an alternative to the need for separate single-level DSAs and interenclave guards, in essence replacing multiple guards and DSAs with a single NSM running the MLS Directory Server, significantly reducing management overhead in terms of personnel and complexity.

4.6.2 Proposed Longer-Term Enhancements

4.6.2.1 Features to Satisfy MISSI MLS DSA Requirements

As these requirements are finalized by the NSA's X33 group, we propose to develop an implementation plan, design, schedule and level-of-effort scoping for phased implementation of features to meet those requirements. In cases when providing such features requires modification of the standard DX500 OpenDirectory DSA, Wang will work with Datacraft Technologies (the DSA provider) to provide this implementation plan, design, and schedule/scoping.

4.6.2.2 ACP-133

When the international ACP-133 standard is finalized, we will define any ACP-133 capabilities required by the US (e.g., DMS) and not supported in the *international* DX500 OpenDirectory implementation, and work with Datacraft to scope the effort of implementing the DMS/US-unique features.

4.6.2.3 X.500 Standard Operational Security Enhancements

There is currently under review in the X.500 community a set of Draft Amendments (DAMs) to the ISO X.500 Standard Support Enhancement of Directory Operational Security. These enhancements include:

1) Integrity of stored data based on digital signatures;
2) Confidentiality of stored data based on encryption;
3) Auditing facilities;
4) Rules-based access control;
5) Context-based access control.

As these DAMs become stable, we propose to develop an implementation plan, design, schedule and level-of-effort scoping for phased implementation of features not already planned for the standard DX500 OpenDirectory DSA, Wang will work with Datacraft Technologies (the DSA provider) to provide this implementation plan, design, and schedule/scoping.

5. PROCESSING RESOURCES

Because it is designed to run on the XTS-300, the MLS Directory Server will be constrained, to some extent, by the processing resources available on that system. This said, we anticipate demonstrating the proof-of-concept system on the new XTS-300 Model 3P1, which has the following system resources:

- 133MHz Pentium CPU
- 32MB memory
- 1 GB hard drive
- 2 Ethernet connections
- dual PCMCIA (FORTEZZA) card reader

If necessary, memory, disk space, and network connections can be increased, though for the proof-of-concept, we do not anticipate this being necessary. One of the benefits, in terms of resource utilization, of the DX500 OpenDirectory architecture is that it does not load into memory, as do ISODE-based X.500 DSAs, but instead inherits the RDBMS's disk-oriented resource utilization. In this way, the MLS Directory Server will minimize memory requirements when compared with many other single-level DSA products, which require large amounts of memory to load the entire directory, and are thus constrained in the number of directory entries they can support.

As the future operational MLS Directory Server is developed, the XTS-300 is also being enhanced further to support multiprocessing (multiple CPU-configuration). In addition, a 166MHz CPU is also being tested, as is the ability to support multiple PCMCIA readers.

Performance and resource utilization data for the DX500 OpenDirectory running against an Ingres database in a Sun SPARC environment can be made available upon request. Performance and resource utilization data for the XTS-300 (running FTP file transfers) and Trusted RUBIX can also be requested, but will be of questionable relevance when trying to anticipate proof-of-concept resource utilization and performance expectations.

APPENDIX C

MLS X.500 Initial Directory Schema

```
schema set attribute dms-attr:2 = {
    name = dmsCreateTimeStamp
    syntax = uTCTime
    single-valued
};

schema set attribute dms-attr:3 = {
    name = dmsAlternateRecipient
    syntax = distinguishedName
};

schema set attribute dms-attr:4 = {
    name = dmsAssociatedOrganization
    syntax = distinguishedName
};

schema set attribute dms-attr:5 = {
    name = dmsAssociatedMI
    syntax = distinguishedName
};

schema set attribute dms-attr:6 = {
    name = dmsAssociatedPLA
    syntax = distinguishedName
};

schema set attribute dms-attr:7 = {
    name = dmsNDNPolicy
    syntax = dmsNDNPolicySyntax              # NDNPolicy
    single-valued
};

# NDNPolicy ::= ENUMERATED {OWNER(0), ORIGINATOR(1), BOTH(2) }

schema set attribute dms-attr:8 = {
    name = dmsMIType
    syntax = dmsMITypeSyntax
    single-valued
};

# dmsMITypeSyntax = ENUMERATED {AIG(0), TYPE(1), CAD(2), TASKFORCE(3) ....}

schema set attribute dms-attr:9 = {
    name = dmsPlaName
    syntax = caseIgnoreString
    single-valued
};
```

```
schema set attribute dms-attr:11 = {
   name = dmsPlaTAREFlag
   syntax = boolean
   single-valued
};

schema set attribute dms-attr:13 = {
   name = dmsPlaMinimizeOverrideFlag
   syntax = boolean
   single-valued
};

schema set attribute dms-attr:14 = {
   name = dmsPlaSectionFlag
   syntax = boolean
   single-valued
};

schema set attribute dms-attr:15 = {
   name = dmsPlaDualRouteFlag
   syntax = boolean
   single-valued

schema set attribute dms-attr:16 = {
   name = dmsPlaServiceOrAgency
   syntax = caseIgnoreString
   single-valued
};

schema set attribute dms-attr:17 = {
   name = dmsPlaPublishFlag
   syntax = boolean
   single-valued
};

schema set attribute dms-attr:19 = {
   name = dmsDodaac
   syntax = caseIgnoreString
};

schema set attribute dms-attr:20 = {
   name = dmsPlaExpirationDate
   syntax = uTCTime
   single-valued
};

schema set attribute dms-attr:21 = {
   name = dmsPlaLongTitle
   syntax = caseIgnoreString
   single-valued
};
schema set attribute dms-attr:22 = {
    name = dmsPlaRIInfo
    syntax = dmsRIParametersSyntax                    # RIParameters
};


# RIParameters ATTRIBUTE-SYNTAX
#     SET     {
#               routingIndicator            [0] RoutingIndicator,
#               rIType                   [1] RIType,
#          rIDeliveryPreference             [2] DeliveryPreference,
#               minimizeFlag                [3] BOOLEAN,
#          sHD                    [5] SpecialHandlingDesig,
#          RIClassification                [6] DestinationClassification}
#
# RoutingIndicator ::= PrintableString (SIZE(7))
# RIType ::= PrintableString (SIZE (1))
#     (FROM ("N" - normal - I
#         "0" - offline - I
#         "P" - part time terminal - I
#         "A" - ADI - ))
#
# Classification ::= PrintableString (SIZE (1))
#     (FROM ("S" - secret - I
#         "C" - confidential - I
#         "R" - restricted - I
#         "E" - unclassified EFTO - I
#         "U" - unclassified - ))
#
# DestinationClassification ::= PrintableString (SIZE (1))
#     (FROM "T" - top secret - I
#         "S" - secret - I
#         "C" - confidential - I
#         "R" - restricted - I
#         "E" - unclassified EFTO - I
#         "U" - unclassified - ))
#
# SpecialHandlingDesig ::= PrintableString

schema set attribute dms-attr:23 = {
    name = dmsPlaActionAddresses
    syntax = caseIgnoreString
};

schema set attribute dms-attr:24 = {
    name = dmsPlaInfoAddresses
    syntax = caseIgnoreString
};
```

```
schema set attribute dms-attr:25 = {
    name = dmsPlaCognizantAuthority
    syntax = caseIgnoreString
    single-valued
};

schema set attribute dms-attr:26 = {
    name = dmsPlaLastRecapDate
    syntax = uTCTime
    single-valued
};

schema set attribute dms-attr:27 = {
    name = dmsPlaRecapDueDate
    syntax = uTCTime
    single-valued
};

schema set attribute dms-attr:28 = {
    name = dmsPlaEffectiveDate
    syntax = uTCTime
    single-valued
};

schema set attribute dms-attr:29 = {
    name = dmsPlaAllowableOriginators
    syntax = caseIgnoreString
};

schema set attribute dms-attr:30 = {
    name = dmsOwningCountry
    syntax = printableString
    single-valued
};

schema set attribute dms-attr:31 = {
    name = dmsPlaRemarks
    syntax = caseIgnoreString
};

schema set attribute dms-attr:35 = {
    name = dmsPlaStateName
    syntax = caseIgnoreString
};

schema set attribute dms-attr:36 = {
    name = dmsPlaProvinceName
    syntax = caseIgnoreString
};
```

```
schema set attribute dms-attr:37 = {
    name = dmsPlaRegionName
    syntax = caseIgnoreString
};

schema set attribute dms-attr:38 = {
    name = dmsPlaEntryClassification
    syntax = dmsClassificationSyntax                # Classification
};

schema set attribute dms-attr:39 = {
    name = dmsPlaNameClassification
    syntax = dmsClassificationSyntax                # Classification
};

schema set attribute dms-attr:40 = {
    name = dmsPlaMinimize
    syntax = boolean
};

schema set attribute dms-attr:41 = {
    name = dmsPrimarySpelling
    syntax = caseIgnoreString
    single-valued
};

schema set attribute dms-attr:42 = {
    name = dmsPlaReplaceFlag
    syntax = boolean
    single-valued
};

schema set attribute dms-attr:43 = {
    name = dmsHostOrganizationalPLA
    syntax = caseIgnoreString
    single-valued
};

schema set attribute dms-attr:45 = {
    name = dmsReleaseAuthorityName
    syntax = caseIgnoreString
};


# Object Classes
schema set oid-prefix dms-objectClass = (2.16.840.1.101.2.2.3);

schema set object-class dms-objectClass:0 = {
    name = dmsEntry
    subclass-of top
                    #must-contain createTimeStamp
};
```

```
};

schema set object-class dms-objectClass:1 = {
   name = dmsSMTPUser
   subclass-of top
   may-contain
      cosineRfc822Mailbox
};

schema set object-class dms-objectClass:2 = {
   name = dmsPOCOrganizationalUnit
   subclass-of organizationalUnit         # mhs-user, dmsSMTPUser,
msp-user-sdns, msp-user-mosaic
   must-contain
      dmsPreferredDelivery,
      dmsOwningCountry
   may-contain
      dmsAssociatedPLA,
      dmsAlternateRecipient,
      dmsBackpointers
};

schema set object-class dms-objectClass:3 = {
   name = dmsPOCOrganizationalPerson
   subclass-of organizationalPerson    # mhs-user, dmsSMTPUser,
msp-user-sdns, msp-user-mosaic
   must-contain
      dmsPreferredDelivery,
      dmsOwningCountry
   may-contain
      dmsAssociatedOrganization,
      dmsAlternateRecipient,
      dmsBackpointers
};

schema set object-class dms-objectClass:4 = {
   name = dmsMl
   subclass-of dmsSMTPUser              # msp-user-sdns, mhs-user,
msp-user-mosaic, mail-list
   must-contain
      commonName,
      mhsDLSubmitPermissions         # mhs-dl-submit-permissions
   may-contain
      dmsNDNPolicy,
      dmsMlType,
      description,
      dmsBackpointers,
      dmsAlternateRecipient
};

schema set object-class dms-objectClass:5 = {
   name = dmsMLA
   subclass-of applicationEntity
};
```

```
schema set object-class dms-objectClass:6 = {
    name = dmsCertificationAuthority
    subclass-of dmsSMTPUser                # ca-mosaic, ca-sdns,
organizationalRole, mhs-user,
                          # msp-user-sdns, msp-user-mosaic
    must-contain
        dmsPreferredDelivery
    may-contain
        dmsBackpointers
};

schema set object-class dms-objectClass:7 = {
    name = dmsADIGateway
    subclass-of applicationEntity          # msp-user-mosaic, mhs-user
    may-contain
        cosineHost
};

schema set object-class dms-objectClass:8 = {
    name = dmsPla
    subclass-of top
    must-contain
        dmsPlaName,
        dmsPlaServiceOrAgency,
        dmsPlaPublishFlag,
                          #dmsPlaEffectiveData,
        dmsOwningCountry
    may-contain
        dmsPlaExpirationDate,
        dmsPlaRemarks
};

schema set object-class dms-objectClass:9 = {
    name = dmsOrganizationalPLA
    subclass-of dmsPla
    must-contain
        dmsPlaMinimize,
        dmsPlaSectionFlag,
        dmsPlaDualRouteFlag,
        dmsPlaMinimizeOverrideFlag,
        dmsPlaTAREFlag
    may-contain
        dmsPlaNameClassification,
        dmsPlaEntryClassification,
        localityName,
        dmsPlaStateName,
        dmsPlaProvinceName,
        dmsPlaRegionName,
        countryName,
        dmsPlaLongTitle,
        dmsDodaac,
        dmsPlaRIInfo,
        dmsAssociatedOrganization
};

schema set object-class dms-objectClass:12 = {
    name = dmsReleaseAuthority
    subclass-of top
    must-contain
        dmsReleaseAuthorityName
    may-contain
        mosaicKMandSigCertificate
        # sdnsUserSignatureCertificate - unknown attribute
};

schema set object-class dms-objectClass:13 = {
    name = dmsPlaCollective
    subclass-of dmsPla
    must-contain
        dmsPlaCognizantAuthority,
        dmsPlaLastRecapDate,
        dmsPlaRecapDueDate
    may-contain
        dmsPlaEntryClassification,
        dmsPlaActionAddresses,
        dmsPlaInfoAddresses,
        dmsPlaAllowableOriginators,
        dmsAssociatedMI
};

schema set object-class dms-objectClass:14 = {
    name = dmsComputer
    subclass-of device
};

schema set object-class dms-objectClass:15 = {
    name = dmsOSIGateway
    subclass-of applicationEntity          # mhs-user, msp-user-mosaic
    may-contain
        cosineHost
};

schema set object-class dms-objectClass:16 = {
    name = dmsAliasOrganizationalUnit
    subclass-of alias
    must-contain
        organizationalUnitName
};
```

```
schema set object-class dms-objectClass:17 = {
    name = dmsAliasOrganizationalPerson
    subclass-of alias
    must-contain
        commonName
};

schema set object-class dms-objectClass:18 = {
    name = dmsAliasOrganizationalRole
    subclass-of alias
    must-contain
        commonName
};

schema set object-class dms-objectClass:19 = {
    name = dmsAliasMI
    subclass-of alias
    must-contain
        commonName
};

schema set object-class dms-objectClass:20 = {
    name = dmsTaskForcePLA
    subclass-of dmsPla
    must-contain
        dmsPlaCognizantAuthority,
        dmsPlaLastRecapDate,
        dmsPlaRecapDueDate
    may-contain
        dmsPlaEntryClassification,
                                # dmsPlaAddresses,
        dmsAssociatedMI
};

schema set object-class dms-objectClass:21 = {
    name = dmsTenantPLA
    subclass-of dmsPla
    must-contain
        dmsHostOrganizationalPLA
    may-contain
        dmsPlaEntryClassification,
        dmsPlaTAREFlag
};

schema set object-class dms-objectClass:22 = {
    name = dmsAlternateSpellingPLA
    subclass-of dmsPla
    must-contain
        dmsPlaReplaceFlag,
        dmsPrimarySpelling
};
```

```
schema set object-class dms-objectClass:23 = {
   name = dmsCadPLA
   subclass-of dmsPla
   must-contain
      dmsPlaCognizantAuthority
   may-contain
      dmsPlaEntryClassification,
      dmsAssociatedMI,
      dmsPlaRIInfo,
      dmsPlaRecapDueDate
};

schema set object-class dms-objectClass:24 = {
   name = dmsPOCOrganizationalRole
   subclass-of
      organizationalRole        # mhs-user, dmsSMTPUser, msp-user-sdns,
msp-user-mosaic
   must-contain
      dmsPreferredDelivery,
      dmsOwningCountry
   may-contain
      dmsAlternateRecipient,
      dmsBackpointers
};
```

APPENDIX D

MLS X.500 Directory Test Plan and Demonstration Scenarios

Wang X.500 MLS Directory Test Plan

1.0 Introduction

The U.S. and Allies are migrating to the use of open systems solutions based on international standards for their messaging, network management, and document interchange systems. Within the U.S., the DoD is in the process of developing a single messaging system for all individual and organizational messaging.. This system, the Defense Messaging System (DMS) is based on the use of the X.400 Message Handling System combined with the Secure Data Network System (SDNS) Message Security Protocol (MSP),and the X.500 Directory System protocols. The combination of these technologies will provide the DoD with messaging, security and Directory services necessary to implement global messaging capabilities. The X.500 Directory System provides an integral part of the DMS solution's infrastructure by providing a place to store and distribute addressing and security information.

Currently, the DMS solution addresses the Unclassified-But-Sensitive (SBU) environment. As DMS evolves to address the secret and top-secret environments, the storage, distribution, and maintenance of classified information becomes a large problem. This document describes test steps that can be used to validate the implementation of the Multi-Level Secure (MLS) X.500 Directory Service running on the Wang XTS-300 trusted platform and holding data at different levels of security classification in a RUBIX trusted MLS data base.

2.0 Test Environment

Application Software	Operating System	Hardware
MLS Directory Server	STOP 4.3.1	XTS-300 • 133 Mhz Pentium • 64 MB RAM • 1 GB Hard Drive • (2) Fortezza Readers
DX500 DSA	SUN OS 4.1.3	SPARCstation-10 • 32 MB Ram • 2 BG Hard drive • External FORTEZZA
DX Explorer DUA	Windows 3.11	Intel Pentium • 133 Mhz CPU • 64 MB RAM • 2 GB Hard Drive • External Fortezza

3.0 Test Plan

2.1 Datacraft port

The Datacraft port to the Wang XTS-300 can be tested prior to integrating the RUBIX database by chaining to a second DSA that contains a database. This step will be done using the Datacraft implementation on jg_sun2. This tests that DSP is working correctly on a single security level.

DAP can be tested by using any available DUA to query the Directory connecting to the Wang and chaining to the information on the sun.

Access control should be tested by populating the Directory with a small portion of the DMS schema and setting ACI information to allow different administrators control over different portions of the data. For instance one administrator should have control over adding in new entries, but a second should have control of the security information. This will simulate the environment of a CAW vs ADUA.

2.2 Fortezza integration

If a DUA that does strong authentication can be located, this can be used to test the DAP Bind using strong authentication. If not, VDA will have to add strong authentication to a test engine or to another DUA.

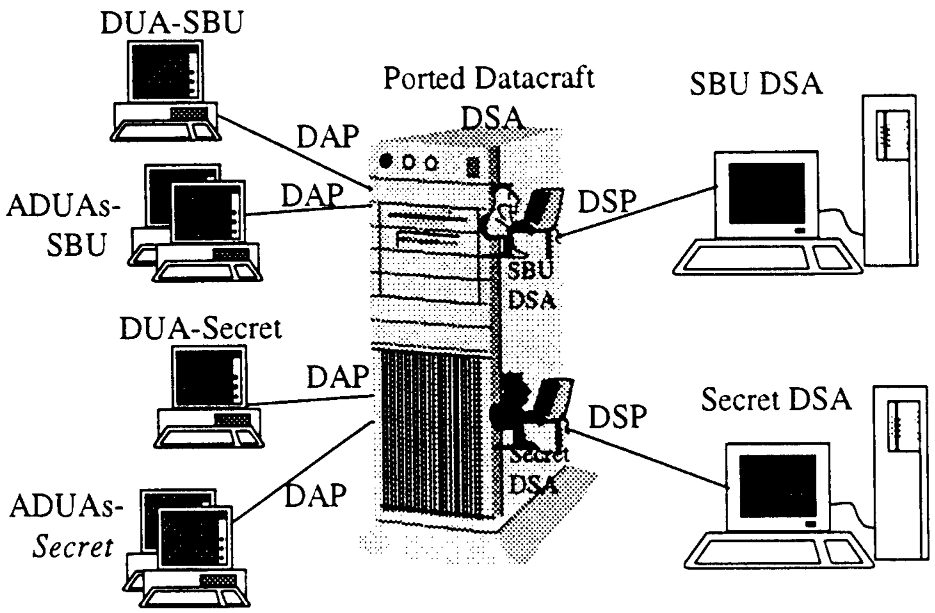
Strong authentication between DSAs will be tested by configuring two copies of the Datacraft DSA to chain with each other. Again the Datacraft can be put on jg_sun2 and on the Wang, or multiple copies of the DSA can be run on the Wang.

2.3 Rubix

Once the RUBIX data base is ported, information can be stored on the Wang and accessed using DAP from any DUA.

The MLS Directory will be configured to be able to chain to two single security level DSAs, one containing SBU data and the other containing only Secret data. The MLS Directory will use different physical network connections to determine the security level of the chained DSA.

Multiple users will access the MLS DSA, in order to show that all access controls work properly for the correct level of authentication. DUAs with administrative capabilities as well as lookup capabilities will be required to demonstrate access controls as well as separation of security classifications.


DUA-SBU
Ported Datacraft DSA
SBU DSA
DAP
ADUAs-SBU
DAP
DSP
SBU DSA
DUA-Secret
DAP
Secret DSA
DSP
Secret DSA
ADUAs-*Secret*
DAP

Demo Scenarios

Directory users	Access privileges	Security level
ADUA1	administrative access to SBU data, excluding security information	SBU
CAW/ADUA1	administrative access to security information for SBU users	SBU
ADUA2	administrative access to SECRET data including security information	SECRET
DUA1	SBU user data	SBU
DUA2	SECRET user data	SECRET
DUA3	outsider	should not get access at all

Scenario 1

DUA3 attempts to bind to the MLS DSA using each of the addresses sequentially for the unclassified, and SECRET DSA.

Each attempt should result in a BIND error

Scenario 2

DUA1 attempts to bind to the MLS DSA using each of the addresses sequentially for the SBU and SECRET DSA.

The bind to the SBU DSA should succeed and DUA1 should be able to browse through the SBU portion of the directory seeing only user data.

The bind to the SECRET DSA should result in a bind error.

Scenario 3

DUA2 attempts to bind to the MLS DSA using each of the addresses sequentially for the SBU and SECRET DSA.

The bind to the SECRET DSA should succeed and DUA2 should be able to browse through the SECRET portion of the directory seeing only user data.

The bind to the SBU DSA should result in a bind error.

Scenario 4

ADUA1 adds a new user, tries to add security information and gets an access control violation.
CAW/ADUA1 adds the security information to the new user ADUA1 just added.
CAW/ADUA1 attempts to add a second new user and gets an access control violation.

Scenario 5

Having successfully done a bind under Scenarios 2 and 3, both DUA1 and DUA2 should browse through the directory looking at whatever information they can. Each user should see different data depending on their security level.

Browsing the Directory Information Tree should seamlessly allow each user to access data on a second DSA. Each user should only see chained information at their own security level.

Scenario 6

DUA1 attempts to read a specific directory entry for a SECRET user. This could be accomplished by attempting to read using the Distinguished Name for the entry containing information about DUA2.

Attempt should fail with an access control error.

Scenario 7

DUA2 attempts to read a specific directory entry for an SBU user. This could be accomplished by attempting to read using the Distinguished Name for the entry containing information about DUA1.

Attempt should fail with an access control error.

Scenario 8

ADUA2 adds a new user and his security information.

The following type of information will need to be configured for each of the DSAs used in the demonstration scenarios.

Configuration Information:

```
jg_sun2:
stack set config = {                                            # startup STACK
    p-addr = {  psap = "PP"                                 # dap PSAP
#               ssap = "SS"                                       # absent means any
#               tsap = "TT"                                       # absent means any
                nsap = rfc1006 "158.189.4.100" port 1900        # zeros mean
this host
             }
    dsp-psap = "DSP"
    cmip-psap = "CMIP"
    ldap-port = 1901
    snmp-port = 1902
    };

dsp set remote-dsa = {
        name = "root"
      p-addr = { psap = "DSP"
                   nsap = rfc1006 "158.189.4.104" port 1900 }
        max-idle-time = 60 }


wang:
stack set config = {                                            # startup STACK
    p-addr = {  psap = "PP"                                 # dap PSAP
#               ssap = "SS"                                       # absent means any
#               tsap = "TT"                                       # absent means any
                nsap = rfc1006 "158.189.4.104" port 1900        # zeros mean
this host
             }
    dsp-psap = "DSP"
    cmip-psap = "CMIP"
    ldap-port = 1901
    snmp-port = 1902
    };

dsp set remote-dsa = {
        name = "sun"
        prefix = <countryName "AU">
        p-addr = { psap = "DSP"
                   nsap = rfc1006 "158.189.4.100" port 1900 }
        max-idle-time = 60 }
```

APPENDIX E

Datacraft DSA/Trusted RUBIX File Listings

DATACRAFT DSA/Trusted RUBIX
SUN OS 4.1.3

```
Sep 26 15:56 1997  listing.3 Page 1

#
# only file on tape
#
-rw-rw-r--   1 154      50       32088064 Oct  7 15:07 1996, rick_rubix.tar

#
# contained w/i rick_rubix.tar
#
-rw-rw-r--154/50        0 Aug 16 08:55 1996 START
-rwxrwxr-x154/50        0 Sep 17 17:22 1996 dsa/
-rwxrwxr-x154/50        0 Aug  5 04:14 1996 dsa/dsa/
-rwxrwxr-x154/50        0 Sep 17 17:14 1996 dsa/dsa/rfc1006/
lrwxrwxrwx154/50        8 Aug 16 08:52 1996 dsa/dsa/rfc1006/dsa.c symbolic link to ../dsa.c
lrwxrwxrwx154/50       12 Aug 16 08:52 1996 dsa/dsa/rfc1006/version.c symbolic link to ../version.c
-rw-rw-r--154/50      920 Aug 26 13:36 1996 dsa/dsa/rfc1006/Makefile
-rw-rw-rw-154/50     2205 Sep 17 17:14 1996 dsa/dsa/rfc1006/.make.state
-rw-rw-r--154/50    84660 Aug 27 18:02 1996 dsa/dsa/rfc1006/dsa.o
-rw-rw-r--154/50     2556 Aug 27 18:02 1996 dsa/dsa/rfc1006/version.o
-rw-rw-r--154/50      464 Aug 27 18:03 1996 dsa/dsa/rfc1006/.nse_depinfo
-rw-rw-r--154/50     1145 Aug 21 12:46 1996 dsa/dsa/Makefile
-r--r--r--154/50    13443 Jul 17 08:52 1996 dsa/dsa/dsa.c
-r--r--r--154/50     1189 Aug  5 03:30 1996 dsa/dsa/version.c
-rwxrwxr-x154/50        0 Aug 27 17:30 1996 dsa/dsa/tcp/
lrwxrwxrwx154/50        8 Aug 16 08:52 1996 dsa/dsa/tcp/dsa.c symbolic link to ../dsa.c
lrwxrwxrwx154/50       12 Aug 16 08:52 1996 dsa/dsa/tcp/version.c symbolic link to ../version.c
-r--r--r--154/50      682 Jul 16 22:20 1996 dsa/dsa/tcp/Makefile
-r--r--r--154/50     4915 Jul 16 22:20 1996 dsa/dsa/tcp/asn1dec.c
-r--r--r--154/50     9301 Jul 18 23:44 1996 dsa/dsa/tcp/tcp_dap.c
-r--r--r--154/50     8545 Jul 16 22:20 1996 dsa/dsa/tcp/tcp_if.c
-rw-rw-r--154/50    46018 Jul 18 00:45 1996 dsa/dsa/tcp/Dapidu_r.c
-rw-rw-rw-154/50     5215 Aug 27 17:30 1996 dsa/dsa/tcp/.make.state
-rwxrwxr-x154/50        0 Jul 18 00:38 1996 dsa/api/
-rwxrwxr-x154/50        0 Aug 27 17:30 1996 dsa/api/src/
lrwxrwxrwx154/50       30 Aug 16 08:52 1996 dsa/api/src/Attr_i.c symbolic link to ../../support/pack-fp/Attr_i.c
lrwxrwxrwx154/50       30 Aug 16 08:52 1996 dsa/api/src/Auth_i.c symbolic link to ../../support/pack-fp/Auth_i.c
lrwxrwxrwx154/50       23 Aug 16 08:52 1996 dsa/api/src/Htcp.c symbolic link to ../stack/network/Htcp.c
lrwxrwxrwx154/50       29 Aug 16 08:52 1996 dsa/api/src/MTS_i.c symbolic link to ../../support/pack-fp/MTS_i.c
lrwxrwxrwx154/50       32 Aug 16 08:52 1996 dsa/api/src/Rupper_i.c symbolic link to ../../support/pack-fp/Rupper_i.c
lrwxrwxrwx154/50       29 Aug 16 08:52 1996 dsa/api/src/Syx_i.c symbolic link to ../../support/pack-fp/Syx_i.c
lrwxrwxrwx154/50        6 Aug 16 08:52 1996 dsa/api/src/dsa.h symbolic link to pack.h
lrwxrwxrwx154/50       26 Aug 16 08:52 1996 dsa/api/src/pAttr.c symbolic link to ../../support/pack/pAttr.c
lrwxrwxrwx154/50       25 Aug 16 08:52 1996 dsa/api/src/pMHS.c symbolic link to ../../support/pack/pMHS.c
lrwxrwxrwx154/50       28 Aug 16 08:52 1996 dsa/api/src/pSyntax.c symbolic link to ../../support/pack/pSyntax.c
-r--r--r--154/50     2216 Jul 16 22:20 1996 dsa/api/src/Make.SCO
-r--r--r--154/50     2750 Jul 16 22:20 1996 dsa/api/src/Make.SUN
-r--r--r--154/50      490 Jul 16 22:20 1996 dsa/api/src/Makefile
-r--r--r--154/50     9553 Jul 18 23:45 1996 dsa/api/src/dsapi.c
-r--r--r--154/50     1457 Jul 16 22:20 1996 dsa/api/src/pack.h
lrwxrwxrwx154/50       31 Aug 16 08:52 1996 dsa/api/src/stackGluei.c symbolic link to ../../oper/stackif/stackGluei.c
-rw-rw-rw-154/50     5981 Aug 26 12:54 1996 dsa/api/src/.make.state
-rwxrwxr-x154/50        0 Jul 18 00:38 1996 dsa/api/include/
lrwxrwxrwx154/50       20 Aug 16 08:52 1996 dsa/api/include/Attr.h symbolic link to ../../include/Attr.h
lrwxrwxrwx154/50       20 Aug 16 08:52 1996 dsa/api/include/Auth.h symbolic link to ../../include/Auth.h
lrwxrwxrwx154/50       25 Aug 16 08:52 1996 dsa/api/include/DapDsp.h symbolic link to ../stack/include/DapDsp.h
lrwxrwxrwx154/50       25 Aug 16 08:52 1996 dsa/api/include/Dapasn.h symbolic link to ../stack/include/Dapasn.h
lrwxrwxrwx154/50       22 Aug 16 08:52 1996 dsa/api/include/Dapidu.h symbolic link to ../../include/Dapidu.h
```

```
Sep 26 15:56 1997  listing.3 Page 2

lrwxrwxrwx154/50       22 Aug 16 08:52 1996 dsa/api/include/Dipidu.h symbolic link to ../../include/Dipidu.h
lrwxrwxrwx154/50       23 Aug 16 08:52 1996 dsa/api/include/Info.h symbolic link to ../stack/include/Info.h
lrwxrwxrwx154/50       19 Aug 16 08:52 1996 dsa/api/include/MTS.h symbolic link to ../../include/MTS.h
lrwxrwxrwx154/50       25 Aug 16 08:52 1996 dsa/api/include/RoseId.h symbolic link to ../stack/include/RoseId.h
lrwxrwxrwx154/50       25 Aug 16 08:52 1996 dsa/api/include/Rupper.h symbolic link to ../stack/include/Rupper.h
lrwxrwxrwx154/50       24 Aug 16 08:52 1996 dsa/api/include/SYNTAXES.h symbolic link to ../../include/SYNTAXES.h
lrwxrwxrwx154/50       19 Aug 16 08:52 1996 dsa/api/include/Syx.h symbolic link to ../../include/Syx.h
lrwxrwxrwx154/50       26 Aug 16 08:52 1996 dsa/api/include/asn1sys.h symbolic link to ../stack/include/asn1sys.h
lrwxrwxrwx154/50       24 Aug 16 08:52 1996 dsa/api/include/queue.h symbolic link to ../stack/include/queue.h
lrwxrwxrwx154/50       26 Aug 16 08:52 1996 dsa/api/include/rstypes.h symbolic link to ../stack/include/rstypes.h
-r--r--r--154/50     1409 Jul 16 22:20 1996 dsa/api/include/ds.h
lrwxrwxrwx154/50       22 Aug 16 08:52 1996 dsa/api/include/Dsp.h symbolic link to ../stack/include/Dsp.h
lrwxrwxrwx154/50        8 Aug 16 08:52 1996 dsa/api/dx500 symbolic link to ../dx500
lrwxrwxrwx154/50        8 Aug 16 08:52 1996 dsa/api/stack symbolic link to ../stack
-r--r--r--154/50      493 Jul 16 22:20 1996 dsa/api/Makefile
-rwxrwxr-x154/50        0 Aug 21 12:45 1996 dsa/api/demo/
lrwxrwxrwx154/50       12 Aug 16 08:52 1996 dsa/api/demo/Makefile symbolic link to Makefile.SUN
-r--r--r--154/50      288 Jul 16 22:20 1996 dsa/api/demo/Makefile.PC
-r--r--r--154/50      276 Jul 16 22:20 1996 dsa/api/demo/Makefile.SCO
-r--r--r--154/50      286 Jul 16 22:20 1996 dsa/api/demo/Makefile.SOLARIS
-r--r--r--154/50      305 Jul 16 22:20 1996 dsa/api/demo/Makefile.SUN
-r--r--r--154/50     9359 Jul 16 22:20 1996 dsa/api/demo/attr.c
-r--r--r--154/50    24921 Jul 16 22:20 1996 dsa/api/demo/conf.c
-r--r--r--154/50     5675 Jul 16 22:20 1996 dsa/api/demo/demo.c
-r--r--r--154/50     1860 Jul 16 22:20 1996 dsa/api/demo/demo.h
-r--r--r--154/50    20799 Jul 16 22:20 1996 dsa/api/demo/reg.c
-rwxrwxr-x154/50        0 Aug 21 12:45 1996 dsa/api/lib/
-rwxrwxr-x154/50        0 Jul 19 00:02 1996 dsa/dua/
lrwxrwxrwx154/50       16 Aug 16 08:52 1996 dsa/dua/version.c symbolic link to ../dsa/version.c
-rw-rw-r--154/50     1146 Aug 21 12:46 1996 dsa/dua/Makefile
-r--r--r--154/50    11325 Jul 19 00:02 1996 dsa/dua/dua.c
-rwxrwxr-x154/50        0 Aug 28 11:37 1996 dsa/dua/rfc1006/
lrwxrwxrwx154/50        8 Aug 16 08:52 1996 dsa/dua/rfc1006/dua.c symbolic link to ../dua.c
lrwxrwxrwx154/50       12 Aug 16 08:52 1996 dsa/dua/rfc1006/version.c symbolic link to ../version.c
-r--r--r--154/50      652 Jul 16 22:20 1996 dsa/dua/rfc1006/Makefile
lrwxrwxrwx154/50       31 Aug 16 08:52 1996 dsa/dua/rfc1006/stackGluei.c symbolic link to ../../oper/stackif/stackGluei.c
-rw-rw-rw-154/50     3069 Aug 27 18:03 1996 dsa/dua/rfc1006/.make.state
-rw-rw-r--154/50    79776 Aug 27 18:03 1996 dsa/dua/rfc1006/dua.o
-rw-rw-r--154/50     2556 Aug 27 18:03 1996 dsa/dua/rfc1006/version.o
-rw-rw-r--154/50      555 Aug 27 18:03 1996 dsa/dua/rfc1006/.nse_depinfo
-rw-rw-r--154/50    47848 Aug 27 18:03 1996 dsa/dua/rfc1006/stackGluei.o
-rwxrwxr-x154/50        0 Aug 27 17:30 1996 dsa/dua/tcp/
lrwxrwxrwx154/50       23 Aug 16 08:52 1996 dsa/dua/tcp/asn1dec.c symbolic link to ../../dsa/tcp/asn1dec.c
lrwxrwxrwx154/50        8 Aug 16 08:52 1996 dsa/dua/tcp/dua.c symbolic link to ...dua.c
lrwxrwxrwx154/50       23 Aug 16 08:52 1996 dsa/dua/tcp/tcp_dap.c symbolic link to ../../dsa/tcp/tcp_dap.c
lrwxrwxrwx154/50       22 Aug 16 08:52 1996 dsa/dua/tcp/tcp_if.c symbolic link to ../../dsa/tcp/tcp_if.c
lrwxrwxrwx154/50       12 Aug 16 08:52 1996 dsa/dua/tcp/version.c symbolic link to ./version.c
-r--r--r--154/50      556 Jul 16 22 20 1996 dsa/dua/tcp/Makefile
-rw-rw-r--154/50    49909 Jul 18 00.45 1996 dsa/dua/tcp/Dapidu_i.c
-rw-rw-rw-154/50     4705 Aug 27 17:30 1996 dsa/dua/tcp/.make.state
-rwxrwxr-x154/50        0 Aug 21 12:59 1996 dsa/include/
-rw-rw-r--154/50     4297 Aug 26 13:14 1996 dsa/include/Make.include
-r--r--r--154/50     1129 Jul 16 22:20 1996 dsa/include/dip.h
-r--r--r--154/50     1423 Jul 16 22:20 1996 dsa include/dsa.h
-r--r--r--154/50     2719 Jul 16 22.20 1996 dsa/include/mgmt.h
-r--r--r--154/50     2995 Jul 29 02.33 1996 dsa include/oper.h
```

```
Sep 26 15:56 1997  listing.3 Page 3

-r--r--r--154/50     4148 Jul 29 02 58 1996 dsa/include/schema h
-r--r--r--154/50     1309 May 16 00 17 1996 dsa/include/stack.h
-r--r--r--154/50     2890 Jul 29 02 58 1996 dsa/include/support.h
-r--r--r--154/50     2693 Jul 28 19 39 1996 dsa/include/trace h
-rw-rw-r--154/50    11328 Jul 18 00 45 1996 dsa/include/Dapidu h
-rw-rw-r--154/50     7902 Aug 27 17 31 1996 dsa/include/Attr h
-rw-rw-r--154/50    10179 Aug 27 17 31 1996 dsa/include/Auth.h
-rw-rw-r--154/50    17096 Aug 27 17 31 1996 dsa/include/MTS h
-rw-rw-r--154/50     6985 Aug 27 17 31 1996 dsa/include/Syx h
-rw-rw-r--154/50      956 Aug 27 17 31 1996 dsa/include/SYNTAXES h
-rw-rw-r--154/50    17948 Jul 31 19 38 1996 dsa/include/Dipidu h
-r--r--r--154/50     2174 Aug  5 04 08 1996 dsa/include/access h
-r--r--r--154/50     1535 Jul 18 23 36 1996 dsa/include/stackif.h
-rw-rw-r--154/50     9886 Jul 31 19 38 1996 dsa/include/Auth.h.OLD
-rwxrwxr-x154/50        0 Jul 18 00 38 1996 dsa/support/
-rwxrwxr-x154/50        0 Sep 17 17 14 1996 dsa/support/misc/
-r--r--r--154/50      356 Jul 16 22 20 1996 dsa/support/misc/Makefile
-r--r--r--154/50     8262 Jul 16 22 20 1996 dsa/support/misc/asnFlatten c
-r--r--r--154/50     2356 Feb 27 01 44 1996 dsa/support/misc/initstack c
-r--r--r--154/50    19292 Jul 30 20 16 1996 dsa/support/misc/lme c
-r--r--r--154/50     1866 Jul 16 22 20 1996 dsa/support/misc/other c
-r--r--r--154/50     9193 Jul 18 00 08 1996 dsa/support/misc/trace c
-r--r--r--154/50     8067 Jul 18 00 08 1996 dsa/support/misc/flatten c
-r--r--r--154/50     2052 Jul 16 22 20 1996 dsa/support/misc/trim.c
-rw-rw-rw-154/50     4929 Aug 27 17 47 1996 dsa/support/misc/ make state
-rw-rw-r--154/50    46904 Aug 27 17 47 1996 dsa/support/misc/asnFlatten o
-rw-rw-r--154/50    61208 Aug 27 17 47 1996 dsa/support/misc/flatten o
-rw-rw-r--154/50      520 Aug 27 17 47 1996 dsa/support/misc/.nse_depinfo
-rw-rw-r--154/50    85476 Aug 27 17 47 1996 dsa/support/misc/lme.o
-rw-rw-r--154/50    39080 Aug 27 17 47 1996 dsa/support/misc/other o
-rw-rw-r--154/50    51320 Aug 27 17 47 1996 dsa/support/misc/trace o
-rw-rw-r--154/50    39340 Aug 27 17 47 1996 dsa/support/misc/trim.o
-rwxrwxr-x154/50        0 Sep 17 17 14 1996 dsa/support/pack/
-r--r--r--154/50      330 Jul 16 22 20 1996 dsa/support/pack/Makefile
-r--r--r--154/50     3811 Jul 16 22 20 1996 dsa/support/pack/pAttr c
-r--r--r--154/50     4195 Jul 16 22 20 1996 dsa/support/pack/pMHS.c
-r--r--r--154/50    11633 Jul 18 23 41 1996 dsa/support/pack/pSyntax c
-rw-rw-rw-154/50     2130 Aug 27 17 46 1996 dsa/support/pack/ make state
-rw-rw-r--154/50    63848 Aug 27 17 46 1996 dsa/support/pack/pSyntax o
-rw-rw-r--154/50    41432 Aug 27 17 46 1996 dsa/support/pack/pAttr o
-rw-rw-r--154/50      259 Aug 27 17 46 1996 dsa/support/pack/ nse_depinfo
-rw-rw-r--154/50    17536 Aug 27 17 46 1996 dsa/support/pack/pMHS.o
-rwxrwxr-x154/50        0 Aug 27 17 31 1996 dsa/support/asn/
lrwxrwxrwx154/50       24 Aug 16 08 52 1996 dsa/support/asn/info.asn symbolic link to   /  /stack/asn/info asn
-rw-rw-r--154/50      898 Aug 21 12 47 1996 dsa/support/asn/Makefile
-r--r--r--154/50     3566 Jul 16 22 20 1996 dsa/support/asn/attr asn
-rw-rw-r--154/50     4756 Aug 21 14 04 1996 dsa/support/asn/auth.asn
-r--r--r--154/50     3926 Jul 16 22 20 1996 dsa/support/asn/basicAC asn
-r--r--r--154/50      841 Jul 16 22 20 1996 dsa/suppor /asn/defs.asn
-r--r--r--154/50      557 Jul 16 22 20 1996 dsa/suppor /asn/fix.awk
-r--r--r--154/50     9898 Jul 16 22 20 1996 dsa/support/asn/mts asn
-r--r--r--154/50     4004 Jul 16 22 20 1996 dsa/support/asn/syx.asn
-rw-rw-r--154/50    27366 Aug 27 17 30 1996 dsa/support/asn/ALL.asn
lrwxrwxrwx154/50       25 Aug 16 08 52 1996 dsa/support asn/upper asn symbolic link to   /  /stack/asn/upper asn
-rwxrwxr-x154/50        0 Sep 17 17 14 1996 dsa/support pack-fp/
-r--r--r--154/50      353 Jul 16 22 20 1996 dsa/support pack-fp/Makefile
```

```
Sep 26 15:56 1997  listing.3 Page 4

-rw-rw-rw-154/50     2054 Aug 27 17 47 1996 dsa/support/pack-fp/ make.state
-rw-rw-r--154/50    27198 Aug 27 17 31 1996 dsa/support/pack-fp/Attr_i c
-rw-rw-r--154/50    38835 Aug 27 17 31 1996 dsa/support/pack-fp/Auth_i.c
-rw-rw-r--154/50    58582 Aug 27 17 31 1996 dsa/support/pack-fp/MTS_i c
-rw-rw-r--154/50    16657 Aug 27 17 31 1996 dsa/support/pack-fp/Rupper_i c
-rw-rw-r--154/50    21283 Aug 27 17 31 1996 dsa/support/pack-fp/Syx_i c
-rw-rw-r--154/50    47208 Aug 27 17 46 1996 dsa/support/pack-fp/Auth_i o
-rw-rw-r--154/50    41920 Aug 27 17 46 1996 dsa/support/pack-fp/Attr_i o
-rw-rw-r--154/50      435 Aug 27 17 47 1996 dsa/support/pack-fp/ nse_depinfo
-rw-rw-r--154/50    67520 Aug 27 17 46 1996 dsa/support/pack-fp/MTS_i o
-rw-rw-r--154/50    22820 Aug 27 17 46 1996 dsa/support/pack-fp/Rupper_i o
-rw-rw-r--154/50    44120 Aug 27 17 47 1996 dsa/support/pack-fp/Syx_i o
-r--r--r--154/50      293 Jul 16 22 20 1996 dsa/support/Makefile
-rwxrwxr-x154/50        0 Sep 17 17 14 1996 dsa/mgmt/
-r--r--r--154/50     3080 Jul 18 00 03 1996 dsa/mgmt/MGMT h
-r--r--r--154/50      549 Jul 16 22 21 1996 dsa/mgmt/Makefile
-r--r--r--154/50    20120 Jul 16 22 21 1996 dsa/mgmt/dumpDN.c
-r--r--r--154/50    37393 Jul 16 22 21 1996 dsa/mgmt/dumpDapidu c
-r--r--r--154/50    24926 Jul 16 22 21 1996 dsa/mgmt/dumpTime c
-r--r--r--154/50    16152 Jul 16 22 21 1996 dsa/mgmt/dumpValue c
-rw-rw-r--154/50    28035 Aug 16 10 03 1996 dsa/mgmt/mInput c
-r--r--r--154/50     9460 Jul 16 22 21 1996 dsa/mgmt/mUtil c
-r--r--r--154/50     8150 Jul 16 22 21 1996 dsa/mgmt/parseDN c
-r--r--r--154/50    39644 Jul 18 00 04 1996 dsa/mgmt/parseDapidu c
-r--r--r--154/50     3368 Jul 16 22 21 1996 dsa/mgmt/parseDip c
-r--r--r--154/50     9076 Apr 29 05 12 1996 dsa/mgmt/parseDop c
-r--r--r--154/50    15173 Jul 16 22 21 1996 dsa/mgmt/parseMhs c
-r--r--r--154/50    16845 Jul 28 19 40 1996 dsa/mgmt/parseSchema c
-r--r--r--154/50    24258 Aug  5 04 09 1996 dsa/mgmt/parseScript c
-r--r--r--154/50     6777 Jul 16 22 21 1996 dsa/mgmt/parseStack c
-r--r--r--154/50     2953 Jul 28 19 40 1996 dsa/mgmt/parseTrace c
-r--r--r--154/50    27951 Jul 16 22 21 1996 dsa/mgmt/parseValue c
-r--r--r--154/50     6287 Jul 28 19 40 1996 dsa/mgmt/parseDsp c
-r--r--r--154/50    23590 Jul 18 00 03 1996 dsa/mgmt/dumpSummary c
-r--r--r--154/50    11748 Aug  5 04 09 1996 dsa/mgmt/parseAccess c
-r--r--r--154/50     5824 Jul 18 00 04 1996 dsa/mgmt/parseAssoc c
-rw-rw-r--154/50    74048 Aug 27 17 44 1996 dsa/mgmt/mUtil o
-r--r--r--154/50     3851 Jul 16 22 21 1996 dsa/mgmt/parseDisp c
-r--r--r--154/50     6480 Jul 17 08 29 1996 dsa/mgmt/parseOper c
-rw-rw-rw-154/50    16247 Aug 27 17 46 1996 dsa/mgmt/ make state
-rw-rw-r--154/50    80816 Aug 27 17 43 1996 dsa/mgmt/dumpDN o
-rw-rw-r--154/50    92752 Aug 27 17 43 1996 dsa/mgmt/dumpDapidu o
-rw-rw-r--154/50     1763 Aug 27 17 46 1996 dsa/mgmt/ nse_depinfo
-rw-rw-r--154/50    84352 Aug 27 17 43 1996 dsa/mgmt/dumpSummary o
-rw-rw-r--154/50    86756 Aug 27 17 43 1996 dsa/mgmt/dumpTime o
-rw-rw-r--154/50    79404 Aug 27 17 43 1996 dsa/mgmt/dumpValue o
-rw-rw-r--154/50    85840 Aug 27 17 44 1996 dsa/mgmt/mInput o
-rw-rw-r--154/50    75204 Aug 27 17 44 1996 dsa/mgmt/parseAccess o
-rw-rw-r--154/50    58048 Aug 27 17 44 1996 dsa/mgmt/parseAssoc o
-rw-rw-r--154/50    67736 Aug 27 17 44 1996 dsa/mgmt/parseDN o
-rw-rw-r--154/50    95040 Aug 27 17 44 1996 dsa/mgmt/parseDapidu o
-rw-rw-r--154/50    63756 Aug 27 17 44 1996 dsa/mgmt/parseDip o
-rw-rw-r--154/50    58264 Aug 27 17 45 1996 dsa/mgmt/parseDisp o
-rw-rw-r--154/50    69504 Aug 27 17 45 1996 dsa/mgmt/parseDsp o
-rw-rw-r--154/50    77340 Aug 27 17 45 1996 dsa/mgmt/parseMhs o
-rw-rw-r--154/50    69108 Aug 27 17 45 1996 dsa/mgmt/parseOper o
```

```
-rw-rw-r--154/50   82180 Aug 27 17:45 1996 dsa/mgmt/parseSchema.o
-rw-rw-r--154/50   84088 Aug 27 17:45 1996 dsa/mgmt/parseScript.o
-rw-rw-r--154/50   71580 Aug 27 17:45 1996 dsa/mgmt/parseStack.o
-rw-rw-r--154/50   55348 Aug 27 17:45 1996 dsa/mgmt/parseTrace.o
-rw-rw-r--154/50   93820 Aug 27 17:46 1996 dsa/mgmt/parseValue.o
-rwxrwxr-x154/50       0 Sep 17 17:14 1996 dsa/schema/
-r--r--r--154/50     377 Jul 28 19:40 1996 dsa/schema/Makefile
-r--r--r--154/50    1953 Jul 29 05:23 1996 dsa/schema/SCHEMA.h
-r--r--r--154/50    9124 Jul 29 05:23 1996 dsa/schema/sAttr.c
-r--r--r--154/50    6187 Jul 28 19:40 1996 dsa/schema/sAttrSet.c
-r--r--r--154/50    9086 Jul 28 19:40 1996 dsa/schema/sNBind.c
-r--r--r--154/50   10390 Jul 28 19:40 1996 dsa/schema/sOClass.c
-r--r--r--154/50    4686 Jul 28 19:40 1996 dsa/schema/sPrefix.c
-r--r--r--154/50    3694 Jul 28 19:40 1996 dsa/schema/sSyntax.c
-r--r--r--154/50   19627 Jul 28 19:40 1996 dsa/schema/sUpper.c
-r--r--r--154/50   11396 Jul 28 19:40 1996 dsa/schema/sUtils.c
-r--r--r--154/50    4483 Jul 28 19:40 1996 dsa/schema/sFlatten.c
-rw-rw-rw-154/50    6033 Aug 27 17:42 1996 dsa/schema/.make.state
-rw-rw-r--154/50   69420 Aug 27 17:41 1996 dsa/schema/sAttr.o
-rw-rw-r--154/50   50220 Aug 27 17:41 1996 dsa/schema/sAttrSet.o
-rw-rw-r--154/50     735 Aug 27 17:42 1996 dsa/schema/.nse_depinfo
-rw-rw-r--154/50   49320 Aug 27 17:41 1996 dsa/schema/sFlatten.o
-rw-rw-r--154/50   55540 Aug 27 17:41 1996 dsa/schema/sNBind.o
-rw-rw-r--154/50   54436 Aug 27 17:41 1996 dsa/schema/sOClass.o
-rw-rw-r--154/50   48068 Aug 27 17:41 1996 dsa/schema/sPrefix.o
-rw-rw-r--154/50   47860 Aug 27 17:42 1996 dsa/schema/sSyntax.o
-rw-rw-r--154/50   61276 Aug 27 17:42 1996 dsa/schema/sUpper.o
-rw-rw-r--154/50   58408 Aug 27 17:42 1996 dsa/schema/sUtils.o
-rwxrwxr-x154/50       0 Sep 17 17:14 1996 dsa/dip/
-rwxrwxr-x154/50       0 Sep 17 17:14 1996 dsa/dip/exec/
-r--r--r--154/50    7034 Jul 16 22:21 1996 dsa/dip/exec/EXEC.h
-r--r--r--154/50     628 Jul 16 22:21 1996 dsa/dip/exec/Makefile
-r--r--r--154/50    9795 Jul 16 22:21 1996 dsa/dip/exec/eAdd.c
-r--r--r--154/50    4515 Jul 16 22:21 1996 dsa/dip/exec/eAttr.c
-r--r--r--154/50   10506 Jul 16 22:21 1996 dsa/dip/exec/eAttrChoice.c
-r--r--r--154/50    2247 Jul 16 22:21 1996 dsa/dip/exec/eCompare.c
-rw-rw-r--154/50   14140 Aug 23 19:01 1996 dsa/dip/exec/eConfig.c
-r--r--r--154/50    4677 Jul 16 22:21 1996 dsa/dip/exec/eDn.c
-r--r--r--154/50   10447 Jul 16 22:21 1996 dsa/dip/exec/eEid.c
-r--r--r--154/50   13243 Jul 18 23:43 1996 dsa/dip/exec/eEntryInfo.c
-r--r--r--154/50    8892 Jul 18 23:43 1996 dsa/dip/exec/eEntryInfoRow.c
-r--r--r--154/50   35173 Jul 18 23:43 1996 dsa/dip/exec/eEntryInfoSet.c
-r--r--r--154/50   11162 Jul 16 22:21 1996 dsa/dip/exec/eFilter.c
-r--r--r--154/50   12930 Jul 16 22:21 1996 dsa/dip/exec/eFilterDivide.c
-r--r--r--154/50   16579 Jul 16 22:21 1996 dsa/dip/exec/eFilterGen.c
-rw-rw-r--154/50   16966 Aug 16 09:41 1996 dsa/dip/exec/eFilterItem.c
-r--r--r--154/50   20075 Jul 16 22:21 1996 dsa/dip/exec/eFilterNorm.c
-r--r--r--154/50   13324 Jul 16 22:21 1996 dsa/dip/exec/eFilterReduce.c
-r--r--r--154/50    9945 Jul 18 00:12 1996 dsa/dip/exec/eFlatten.c
-r--r--r--154/50    8068 Jul 16 22:21 1996 dsa/dip/exec/eFragment.c
-r--r--r--154/50    4389 Jul 18 23:42 1996 dsa/dip/exec/eList.c
-r--r--r--154/50    4482 Jul 16 22:21 1996 dsa/dip/exec/eMain.c
-r--r--r--154/50    5775 Jul 16 22:21 1996 dsa/dip/exec/eModDn.c
-r--r--r--154/50    8012 Jul 16 22:21 1996 dsa/dip/exec/eModify.c
-r--r--r--154/50   20189 Jul 18 23:42 1996 dsa/dip/exec/eNavigate.c
-r--r--r--154/50    2223 Jul 16 22:21 1996 dsa/dip/exec/eRead.c
```

```
-r--r--r--154/50    4537 Jul 16 22:21 1996 dsa/dip/exec/eRemove.c
-r--r--r--154/50   16089 Jul 16 22:21 1996 dsa/dip/exec/eSubtree.c
-r--r--r--154/50   18953 Jul 31 06:32 1996 dsa/dip/exec/eUpdate.c
-r--r--r--154/50   13785 Jul 16 22:21 1996 dsa/dip/exec/eUtils.c
-rw-rw-rw-154/50   23336 Aug 27 17:59 1996 dsa/dip/exec/.make.state
-rw-rw-r--154/50   52424 Aug 27 17:56 1996 dsa/dip/exec/eMain.o
-rw-rw-r--154/50   69776 Aug 27 17:56 1996 dsa/dip/exec/eNavigate.o
-rw-rw-r--154/50    2857 Aug 27 17:59 1996 dsa/dip/exec/.nse_depinfo
-rw-rw-r--154/50   49948 Aug 27 17:56 1996 dsa/dip/exec/eCompare.o
-rw-rw-r--154/50   49600 Aug 27 17:56 1996 dsa/dip/exec/eRead.o
-rw-rw-r--154/50   51880 Aug 27 17:56 1996 dsa/dip/exec/eList.o
-rw-rw-r--154/50   60380 Aug 27 17:56 1996 dsa/dip/exec/eSubtree.o
-rw-rw-r--154/50   57132 Aug 27 17:57 1996 dsa/dip/exec/eFilter.o
-rw-rw-r--154/50   75744 Aug 27 17:57 1996 dsa/dip/exec/eFilterReduce.o
-rw-rw-r--154/50   62216 Aug 27 17:57 1996 dsa/dip/exec/eFilterDivide.o
-rw-rw-r--154/50   63096 Aug 27 17:57 1996 dsa/dip/exec/eFilterGen.o
-rw-rw-r--154/50   63012 Aug 27 17:57 1996 dsa/dip/exec/eFilterNorm.o
-rw-rw-r--154/50   64896 Aug 27 17:57 1996 dsa/dip/exec/eFilterItem.o
-rw-rw-r--154/50   57468 Aug 27 17:57 1996 dsa/dip/exec/eAdd.o
-rw-rw-r--154/50   54476 Aug 27 17:57 1996 dsa/dip/exec/eModDn.o
-rw-rw-r--154/50   55380 Aug 27 17:58 1996 dsa/dip/exec/eModify.o
-rw-rw-r--154/50   53680 Aug 27 17:58 1996 dsa/dip/exec/eRemove.o
-rw-rw-r--154/50   68492 Aug 27 17:58 1996 dsa/dip/exec/eUpdate.o
-rw-rw-r--154/50   57360 Aug 27 17:58 1996 dsa/dip/exec/eEntryInfo.o
-rw-rw-r--154/50   55676 Aug 27 17:58 1996 dsa/dip/exec/eEntryInfoRow.o
-rw-rw-r--154/50   81768 Aug 27 17:58 1996 dsa/dip/exec/eEntryInfoSet.o
-rw-rw-r--154/50   50840 Aug 27 17:58 1996 dsa/dip/exec/eAttr.o
-rw-rw-r--154/50   58108 Aug 27 17:58 1996 dsa/dip/exec/eAttrChoice.o
-rw-rw-r--154/50   52608 Aug 27 17:59 1996 dsa/dip/exec/eDn.o
-rw-rw-r--154/50   56144 Aug 27 17:59 1996 dsa/dip/exec/eFragment.o
-rw-rw-r--154/50   55764 Aug 27 17:59 1996 dsa/dip/exec/eEid.o
-rw-rw-r--154/50   61764 Aug 27 17:59 1996 dsa/dip/exec/eConfig.o
-rw-rw-r--154/50   72472 Aug 27 17:59 1996 dsa/dip/exec/eFlatten.o
-rw-rw-r--154/50   77272 Aug 27 17:59 1996 dsa/dip/exec/eUtils.o
-rwxrwxr-x154/50       0 Sep 17 17:14 1996 dsa/dip/asn/
-rw-rw-r--154/50     770 Aug 16 09:27 1996 dsa/dip/asn/Makefile
-r--r--r--154/50     844 Jul 16 22:21 1996 dsa/dip/asn/defs.asn
-r--r--r--154/50    8603 Jul 16 22:21 1996 dsa/dip/asn/dipidu.asn
-rw-rw-rw-154/50     119 Aug 21 12:44 1996 dsa/dip/asn/.make.state
-rwxrwxr-x154/50       0 Aug 21 15:17 1996 dsa/dip/ingres/
-r--r--r--154/50    1249 Jul 16 22:21 1996 dsa/dip/ingres/INGRES.h
-r--r--r--154/50     410 Jul 16 22:21 1996 dsa/dip/ingres/Makefile
-r--r--r--154/50    9526 Jul 16 22:21 1996 dsa/dip/ingres/tAlias.sc
-rw-rw-r--154 50    2613 Aug 23 19:02 1996 dsa/dip/ingres/tAttr.sc
-r--r--r--154/50    9708 Jul 16 22:21 1996 dsa/dip/ingres/tBlob.sc
-r--r--r--154/50   13641 Jul 16 22:21 1996 dsa/dip/ingres/tDit.sc
-r--r--r--154/50   12132 Jul 16 22:21 1996 dsa/dip/ingres/tEntry.sc
-r--r--r--154/50    4029 Jul 16 22:21 1996 dsa/dip/ingres/tInfo.sc
-r--r--r--154/50    8510 Jul 16 22:21 1996 dsa/dip/ingres/tName.sc
-r--r--r--154/50   10672 Jul 16 22:21 1996 dsa/dip/ingres/tSearch.sc
-r--r--r--154/50   10622 Jul 16 22:21 1996 dsa/dip/ingres/tTree.sc
-r--r--r--154/50    6588 Jul 16 22:21 1996 dsa/dip/ingres/tUtils.sc
-rw-rw-rw-154/50    9284 Aug 21 15:17 1996 dsa/dip/ingres/.make.state
-rw-rw-r--154/50   25508 Aug 21 15:15 1996 dsa/dip/ingres/tDit.c
-rw-rw-r--154/50   23923 Aug 21 15:15 1996 dsa/dip/ingres/tSearch.c
-rw-rw-r--154/50   18750 Aug 21 15:15 1996 dsa/dip/ingres/tTree.c
```

```
-rw-rw-r--154/50   14876 Aug 21 15 15 1996 dsa/dip/ingres/tName c
-rw-rw-r--154/50   21994 Aug 21 15 15 1996 dsa/dip/ingres/tEntry c
-rw-rw-r--154/50   16865 Aug 21 15 15 1996 dsa/dip/ingres/tBlob c
-rw-rw-r--154/50    5785 Aug 21 15 15 1996 dsa/dip/ingres/tInfo c
-rw-rw-r--154/50   17520 Aug 21 15 15 1996 dsa/dip/ingres/tAlias c
-rw-rw-r--154/50    5346 Aug 21 15 15 1996 dsa/dip/ingres/tAttr c
-rw-rw-r--154/50    7510 Aug 21 15 15 1996 dsa/dip/ingres/tUtils c
-rw-rw-r--154/50   77828 Aug 21 15 15 1996 dsa/dip/ingres/tDit o
-rw-rw-r--154/50     994 Aug 21 15 17 1996 dsa/dip/ingres/ nse_depinfo
-rw-rw-r--154/50   71680 Aug 21 15 16 1996 dsa/dip/ingres/tTree o
-rw-rw-r--154/50   78264 Aug 21 15 16 1996 dsa/dip/ingres/tSearch o
-rw-rw-r--154/50   65532 Aug 21 15 16 1996 dsa/dip/ingres/tName o
-rw-rw-r--154/50   74212 Aug 21 15 16 1996 dsa/dip/ingres/tEntry o
-rw-rw-r--154/50   67644 Aug 21 15 16 1996 dsa/dip/ingres/tBlob o
-rw-rw-r--154/50   52960 Aug 21 15 16 1996 dsa/dip/ingres/tInfo o
-rw-rw-r--154/50   69624 Aug 21 15 16 1996 dsa/dip/ingres/tAlias o
-rw-rw-r--154/50   53076 Aug 21 15 16 1996 dsa/dip/ingres/tAttr o
-rw-rw-r--154/50   55760 Aug 21 15 17 1996 dsa/dip/ingres/tUtils o
-rwxrwxr-x154/50       0 Sep 17 17 14 1996 dsa/dip/include/
-rw-rw-r--154/50    9367 Aug 23 19 15 1996 dsa/dip/include/DIPINT h
-r--r--r--154/50     364 Jul 16 22 21 1996 dsa/dip/include/Makefile
-r--r--r--154/50    2247 Jul 16 22 21 1996 dsa/dip/include/makeTableHeaders
-rw-r--r--154/50     646 Jul 31 19 48 1996 dsa/dip/include/zalias c
-rw-r--r--154/50     691 Jul 31 19 48 1996 dsa/dip/include/zattr c
-rw-r--r--154/50     733 Jul 31 19 48 1996 dsa/dip/include/zblob c
-rw-r--r--154/50     672 Jul 31 19 48 1996 dsa/dip/include/zdit c
-rw-r--r--154/50     940 Feb 18 22 12 1996 dsa/dip/include/zdit sh
-rw-r--r--154/50    1004 Feb 18 22 12 1996 dsa/dip/include/zsearch sh
-rw-r--r--154/50    1011 Feb 18 22 12 1996 dsa/dip/include/ztree sh
-rw-r--r--154/50     896 Feb 18 22 12 1996 dsa/dip/include/zname sh
-rw-r--r--154/50     979 Feb 18 22 12 1996 dsa/dip/include/zentry sh
-rw-r--r--154/50    1013 Feb 18 22 12 1996 dsa/dip/include/zblob sh
-rw-r--r--154/50     783 Feb 18 22 12 1996 dsa/dip/include/zinfo sh
-rw-r--r--154/50     864 Feb 18 22 12 1996 dsa/dip/include/zalias sh
-rw-r--r--154/50     973 Feb 18 22 12 1996 dsa/dip/include/zattr sh
-rw-r--r--154/50     722 Jul 31 19 48 1996 dsa/dip/include/zentry c
-rw-r--r--154/50     610 Jul 31 19 48 1996 dsa/dip/include/zinfo c
-rw-r--r--154/50     685 Jul 31 19 48 1996 dsa/dip/include/zname c
-rw-r--r--154/50     708 Jul 31 19 48 1996 dsa/dip/include/zsearch c
-rw-r--r--154/50     703 Jul 31 19 48 1996 dsa/dip/include/ztree c
-rw-rw-rw-154/50     147 Aug 26 12 53 1996 dsa/dip/include/ make state
-rw-rw-r--154/50    1214 Aug 23 15 13 1996 dsa/dip/include/Z
-rwxrwxr-x154/50       0 Sep 17 17 14 1996 dsa/dip/mono/
-r--r--r--154/50     309 Jul 16 22 21 1996 dsa/dip/mono/Makefile
-r--r--r--154/50    6409 Jul 18 00 13 1996 dsa/dip/mono/dipMono c
-rw-rw-rw-154/50    1474 Aug 27 18 00 1996 dsa/dip/mono/ make state
-rw-rw-r--154/50   69404 Aug 27 17 59 1996 dsa/dip/mono/dipMono o
-rw-rw-r--154/50     101 Aug 27 17 59 1996 dsa/dip/mono/ nse_depinfo
-rw-rw-r--154/50     271 Aug 24 19 30 1996 dsa/dip/Makefile
-rw-rw-rw-154/50    1640 Sep 17 17 14 1996 dsa/dip/ make state
-rwxrwxr-x154/50       0 Sep 17 17 14 1996 dsa/dip/rubix/
-rw-rw-r--154/50   27108 Aug 26 13 07 1996 dsa/dip/rubix/x
-rw-rw-r--154/50     424 Aug 24 19 10 1996 dsa/dip/rubix/Makefile
-rw-rw-r--154/50    3718 Aug 27 14 42 1996 dsa/dip/rubix/tAttr e
-rw-rw-r--154/50   12119 Aug 28 11 30 1996 dsa/dip/rubix/tBlob e
-rw-rw-r--154/50   15122 Aug 28 11 31 1996 dsa/dip/rubix/tDit e
```

```
-rw-rw-r--154/50   12895 Aug 28 11 31 1996 dsa/dip/rubix/tEntry e
-rw-rw-r--154/50    4053 Aug 24 16 47 1996 dsa/dip/rubix/tInfo e
-rw-rw-r--154/50    9671 Aug 28 11 32 1996 dsa/dip/rubix/tName e
-rw-rw-r--154/50   12962 Aug 27 15 35 1996 dsa/dip/rubix/tSearch e
-rw-rw-r--154/50   11508 Aug 28 11 33 1996 dsa/dip/rubix/tTree e
-rw-rw-r--154/50    9434 Aug 27 15 06 1996 dsa/dip/rubix/tUtils e
-rw-rw-r--154/50    1120 Aug 26 13 19 1996 dsa/dip/rubix/RUBIX h
-rw-rw-rw-154/50   12582 Aug 28 11 36 1996 dsa/dip/rubix/ make state
-rw-rw-r--154/50    5588 Aug 27 18 00 1996 dsa/dip/rubix/tInfo c
-rw-rw-r--154/50   10407 Aug 28 11 29 1996 dsa/dip/rubix/tAlias e
-rw-rw-r--154/50    7615 Aug 27 18 00 1996 dsa/dip/rubix/tAttr c
-rw-rw-r--154/50   20212 Aug 28 11 34 1996 dsa/dip/rubix/tName c
-rw-rw-r--154/50     609 Aug 23 18 38 1996 dsa/dip/rubix/NOTES
-rw-rw-r--154/50     644 Aug 26 13 12 1996 dsa/dip/rubix/where awk
-rw-rw-r--154/50    5588 Aug 27 18 00 1996 dsa/dip/rubix/tInfo pc
-rw-rw-r--154/50   26928 Aug 28 11 34 1996 dsa/dip/rubix/tBlob c
-rw-rw-r--154/50    7615 Aug 27 18 00 1996 dsa/dip/rubix/tAttr pc
-rw-rw-r--154/50   10406 Aug 27 18 00 1996 dsa/dip/rubix/tUtils c
-rw-rw-r--154/50   10406 Aug 27 18 00 1996 dsa/dip/rubix/tUtils pc
-rw-rw-r--154/50   27577 Aug 28 11 34 1996 dsa/dip/rubix/tTree c
-rw-rw-r--154/50   20080 Aug 28 11 34 1996 dsa/dip/rubix/tName pc
-rw-rw-r--154/50   33930 Aug 28 11 34 1996 dsa/dip/rubix/tDit c
-rw-rw-r--154/50   24236 Aug 28 11 34 1996 dsa/dip/rubix/tAlias c
-rw-rw-r--154/50   24104 Aug 28 11 34 1996 dsa/dip/rubix/tAlias pc
-rw-rw-r--154/50   21696 Aug 27 18 00 1996 dsa/dip/rubix/tSearch c
-rw-rw-r--154/50   21696 Aug 27 18 00 1996 dsa/dip/rubix/tSearch pc
-rw-rw-r--154/50   26796 Aug 28 11 34 1996 dsa/dip/rubix/tBlob pc
-rw-rw-r--154/50   27247 Aug 28 11 34 1996 dsa/dip/rubix/tTree pc
-rw-rw-r--154/50   27729 Aug 28 11 34 1996 dsa/dip/rubix/tEntry c
-rw-rw-r--154/50    1549 Aug 26 11 11 1996 dsa/dip/rubix/t c
lrwxrwxrwx154/50      26 Aug 26 11 49 1996 dsa/dip/rubix Make include symbolic link to        include Make include
-rw-rw-r--154/50   27597 Aug 28 11 34 1996 dsa/dip/rubix/tEntry pc
-rw-rw-r--154/50   63844 Aug 27 18 00 1996 dsa/dip/rubix/tInfo o
-rwxrwxr-x154/50  155648 Aug 26 11 11 1996 dsa/dip/rubix/t
-rw-rw-r--154/50   33600 Aug 28 11 34 1996 dsa/dip/rubix/tDit pc
-rw-rw-r--154/50   69488 Aug 27 18 00 1996 dsa/dip/rubix/tAttr o
-rw-rw-r--154/50   68940 Aug 27 18 00 1996 dsa/dip/rubix/tUtils o
-rw-rw-r--154/50    1194 Aug 28 11 35 1996 dsa/dip/rubix/ nse_depinfo
-rw-rw-r--154/50   90740 Aug 28 11 34 1996 dsa/dip/rubix/tName o
-rw-rw-r--154/50   98192 Aug 28 11 34 1996 dsa/dip/rubix/tAlias o
-rw-rw-r--154/50   89268 Aug 27 18 01 1996 dsa/dip/rubix/tSearch o
-rw-rw-r--154/50  102484 Aug 28 11 35 1996 dsa/dip/rubix/tBlob o
-rw-rw-r--154/50  104292 Aug 28 11 35 1996 dsa/dip/rubix/tTree o
-rw-rw-r--154/50  102436 Aug 28 11 35 1996 dsa/dip/rubix/tEntry o
-rw-rw-r--154/50  105872 Aug 28 11 35 1996 dsa/dip/rubix/tDit o
-rw-rw-r--154/50     765 Sep 17 17 14 1996 dsa/dip/ nse_depinfo
-rwxrwxr-x154/50       0 Sep 17 17 14 1996 dsa/oper
-rwxrwxr-x154/50       0 Sep 17 17 14 1996 dsa/oper/local
-r--r--r--154/50    5287 Aug  5 03 06 1996 dsa/oper/local/LOCAL h
-r--r--r--154/50     422 Jul 30 00 53 1996 dsa/oper/local/Makefile
-r--r--r--154/50    9649 Aug  5 03 06 1996 dsa/oper/local/oAdd c
-r--r--r--154/50    3673 Jul 30 00 53 1996 dsa/oper/local/oBlock c
-r--r--r--154/50    8354 Jul 30 00 53 1996 dsa/oper/local/oCompare c
-r--r--r--154/50    8280 Aug  1 18 55 1996 dsa/oper/local/oList c
-r--r--r--154/50    5426 Jul 30 00 53 1996 dsa/oper/local/oMain c
-r--r--r--154/50    8541 Jul 30 00 53 1996 dsa/oper/local/oModDn c
```

```
-r--r--r--154/50    12637 Jul 30 20:42 1996 dsa/oper/local/oModify.c
-r--r--r--154/50     4756 Jul 30 00:53 1996 dsa/oper/local/oNav.c
-r--r--r--154/50     9035 Jul 30 00:53 1996 dsa/oper/local/oRead.c
-r--r--r--154/50     7609 Jul 30 00:53 1996 dsa/oper/local/oRemove.c
-r--r--r--154/50    12521 Aug  1 18:55 1996 dsa/oper/local/oSearch.c
-r--r--r--154/50     7248 Jul 30 00:53 1996 dsa/oper/local/oUtils.c
-rw-rw-rw-154/50     8423 Aug 27 17:40 1996 dsa/oper/local/.make.state
-rw-rw-r--154/50    58560 Aug 27 17:38 1996 dsa/oper/local/oAdd.o
-rw-rw-r--154/50    52032 Aug 27 17:38 1996 dsa/oper/local/oBlock.o
-rw-rw-r--154/50     1198 Aug 27 17:39 1996 dsa/oper/local/.nse_depinfo
-rw-rw-r--154/50    57268 Aug 27 17:38 1996 dsa/oper/local/oCompare.o
-rw-rw-r--154/50    56740 Aug 27 17:39 1996 dsa/oper/local/oList.o
-rw-rw-r--154/50    55504 Aug 27 17:39 1996 dsa/oper/local/oMain.o
-rw-rw-r--154/50    57812 Aug 27 17:39 1996 dsa/oper/local/oModDn.o
-rw-rw-r--154/50    61332 Aug 27 17:39 1996 dsa/oper/local/oModify.o
-rw-rw-r--154/50    52032 Aug 27 17:39 1996 dsa/oper/local/oNav.o
-rw-rw-r--154/50    58204 Aug 27 17:39 1996 dsa/oper/local/oRead.o
-rw-rw-r--154/50    56324 Aug 27 17:39 1996 dsa/oper/local/oRemove.o
-rw-rw-r--154/50    60356 Aug 27 17:39 1996 dsa/oper/local/oSearch.o
-rw-rw-r--154/50    55328 Aug 27 17:39 1996 dsa/oper/local/oUtils.o
-rwxrwxr-x154/50        0 Sep 17 17:14 1996 dsa/oper/user/
-r--r--r--154/50      366 Jul 16 22:21 1996 dsa/oper/user/Makefile
-r--r--r--154/50     3387 Jul 16 22:21 1996 dsa/oper/user/USER.h
-r--r--r--154/50     4668 Jul 16 22:21 1996 dsa/oper/user/uAbandon.c
-r--r--r--154/50     3450 Jul 16 22:21 1996 dsa/oper/user/uAbort.c
-r--r--r--154/50    11905 Jul 28 19:40 1996 dsa/oper/user/uBind.c
-r--r--r--154/50    10759 Jul 17 08:27 1996 dsa/oper/user/uBlock.c
-r--r--r--154/50     8116 Jul 16 22:21 1996 dsa/oper/user/uMain.c
-r--r--r--154/50     8487 Apr  1 00:15 1996 dsa/oper/user/uStack.c
-r--r--r--154/50     5189 Jul 16 22:21 1996 dsa/oper/user/uUtils.c
-rw-rw-rw-154/50     4545 Aug 27 17:38 1996 dsa/oper/user/.make.state
-rw-rw-r--154/50    50980 Aug 27 17:37 1996 dsa/oper/user/uAbandon.o
-rw-rw-r--154/50    49356 Aug 27 17:38 1996 dsa/oper/user/uAbort.o
-rw-rw-r--154/50      600 Aug 27 17:38 1996 dsa/oper/user/.nse_depinfo
-rw-rw-r--154/50    65432 Aug 27 17:38 1996 dsa/oper/user/uBind.o
-rw-rw-r--154/50    74048 Aug 27 17:38 1996 dsa/oper/user/uBlock.o
-rw-rw-r--154/50    71904 Aug 27 17:38 1996 dsa/oper/user/uMain.o
-rw-rw-r--154/50    51676 Aug 27 17:38 1996 dsa/oper/user/uUtils.o
-rwxrwxr-x154/50        0 Jul 30 20:43 1996 dsa/oper/include/
-r--r--r--154/50     3431 Jul 28 19:40 1996 dsa/oper/include/OPER.h
-rwxrwxr-x154/50        0 Sep 17 17:14 1996 dsa/oper/main/
-r--r--r--154/50      322 Jul 16 22:21 1996 dsa/oper/main/Makefile
-r--r--r--154/50     4742 Jul 16 22:21 1996 dsa/oper/main/operMain.c
-rw-rw-rw-154/50     1202 Aug 27 17:41 1996 dsa/oper/main/.make.state
-rw-rw-r--154/50    50416 Aug 27 17:41 1996 dsa/oper/main/operMain.o
-rw-rw-r--154/50      102 Aug 27 17:41 1996 dsa/oper/main/.nse_depinfo
-r--r--r--154/50      267 Jul 16 22:21 1996 dsa/oper/Makefile
-rwxrwxr-x154/50        0 Sep 17 17:14 1996 dsa/oper/remote/
-r--r--r--154/50      378 Jul 16 22:21 1996 dsa/oper/remote/Makefile
-r--r--r--154/50     4655 Jul 29 05:22 1996 dsa/oper/remote/REMOTE.h
-r--r--r--154/50    10034 Jul 18 23:38 1996 dsa/oper/remote/rAnalyse.c
-r--r--r--154/50     5520 Jul 16 22:21 1996 dsa/oper/remote/rBlock.c
-r--r--r--154/50    11980 Jul 28 19:40 1996 dsa/oper/remote/rDsa.c
-r--r--r--154/50     6648 Jul 16 22:21 1996 dsa/oper/remote/rMain.c
-r--r--r--154/50    11311 Jul 18 23:39 1996 dsa/oper/remote/rPrefix.c
-r--r--r--154/50    19528 Jul 28 19:40 1996 dsa/oper/remote/rService.c
```

```
-r--r--r--154/50     3383 Jul 16 22:21 1996 dsa/oper/remote/rUtils.c
-rw-rw-rw-154/50     5333 Aug 27 17:41 1996 dsa/oper/remote/.make.state
-rw-rw-r--154/50    58036 Aug 27 17:40 1996 dsa/oper/remote/rAnalyse.o
-rw-rw-r--154/50    53248 Aug 27 17:40 1996 dsa/oper/remote/rBlock.o
-rw-rw-r--154/50      702 Aug 27 17:40 1996 dsa/oper/remote/.nse_depinfo
-rw-rw-r--154/50    61532 Aug 27 17:40 1996 dsa/oper/remote/rDsa.o
-rw-rw-r--154/50    69308 Aug 27 17:40 1996 dsa/oper/remote/rMain.o
-rw-rw-r--154/50    76068 Aug 27 17:40 1996 dsa/oper/remote/rPrefix.o
-rw-rw-r--154/50    75048 Aug 27 17:40 1996 dsa/oper/remote/rService.o
-rw-rw-r--154/50    51664 Aug 27 17:40 1996 dsa/oper/remote/rUtils.o
-rwxrwxr-x154/50        0 Sep 17 17:14 1996 dsa/oper/stackif/
-r--r--r--154/50      374 Jul 18 23:37 1996 dsa/oper/stackif/Makefile
-r--r--r--154/50     1722 Jul 18 23:37 1996 dsa/oper/stackif/STACKIF.h
-r--r--r--154/50    15768 Jul 18 23:37 1996 dsa/oper/stackif/dispSlave.c
-r--r--r--154/50    22583 Jul 18 23:37 1996 dsa/oper/stackif/stackGlue.c
-r--r--r--154/50     8149 Jul 18 23:37 1996 dsa/oper/stackif/stackGluei.c
-r--r--r--154/50     3751 Jul 18 23:37 1996 dsa/oper/stackif/stackInit.c
-r--r--r--154/50     8612 Jul 18 23:37 1996 dsa/oper/stackif/stackMain.c
-rw-rw-rw-154/50     5103 Aug 27 17:37 1996 dsa/oper/stackif/.make.state
-rw-rw-r--154/50    75428 Aug 27 17:37 1996 dsa/oper/stackif/dispSlave.o
-rw-rw-r--154/50    72408 Aug 27 17:37 1996 dsa/oper/stackif/stackGlue.o
-rw-rw-r--154/50      516 Aug 27 17:37 1996 dsa/oper/stackif/.nse_depinfo
-rw-rw-r--154/50    47848 Aug 27 17:37 1996 dsa/oper/stackif/stackGluei.o
-rw-rw-r--154/50    54972 Aug 27 17:37 1996 dsa/oper/stackif/stackInit.o
-rw-rw-r--154/50    71728 Aug 27 17:37 1996 dsa/oper/stackif/stackMain.o
-rwxrwxr-x154/50        0 Aug 24 19:44 1996 dsa/oper/asn/
-r--r--r--154/50      873 Jul 16 22:21 1996 dsa/oper/asn/Makefile
-rw-r--r--154/50      827 May  1 22:22 1996 dsa/oper/asn/defs.asn
-r--r--r--154/50     5491 Jul 16 22:21 1996 dsa/oper/asn/dapidu.asn
lrwxrwxrwx154/50       26 Aug 16 08:53 1996 dsa/oper/asn/roseId.asn symbolic link to ../../stack/asn/roseId.asn
lrwxrwxrwx154/50       23 Aug 16 08:53 1996 dsa/oper/asn/dap.asn symbolic link to ../../stack/asn/dap.asn
lrwxrwxrwx154/50       26 Aug 16 08:53 1996 dsa/oper/asn/dapdsp.asn symbolic link to ../../stack/asn/dapdsp.asn
lrwxrwxrwx154/50       23 Aug 16 08:53 1996 dsa/oper/asn/dsp.asn symbolic link to ../../stack/asn/dsp.asn
lrwxrwxrwx154/50       25 Aug 16 08:53 1996 dsa/oper/asn/upper.asn symbolic link to ../../stack/asn/upper asn
lrwxrwxrwx154/50       24 Aug 16 08:53 1996 dsa/oper/asn/info.asn symbolic link to ../../stack/asn/info.asn
-rw-rw-rw-154/50     1820 Sep 17 17:14 1996 dsa/oper/.make.state
-rw-rw-r--154/50      936 Sep 17 17:14 1996 dsa/oper/.nse_depinfo
-rwxrwxr-x154/50        0 Aug 28 11:37 1996 dsa/scripts/
-rwxrwxr-x154/50        0 Aug 28 11:37 1996 dsa/scripts/init/
-r--r--r--154/50      401 Jul 16 22:21 1996 dsa/scripts/init/Makefile
-r--r--r--154/50      864 Jul 31 05:43 1996 dsa/scripts/init/init
-r--r--r--154/50    14902 Jul 16 22:21 1996 dsa/scripts/init/init.attr
-r--r--r--154/50    14440 Jul 16 22:21 1996 dsa/scripts/init/init.cosine
-r--r--r--154/50    11283 Jul 16 22:21 1996 dsa/scripts/init/init.dms
-r--r--r--154/50     4948 Jul 16 22:21 1996 dsa/scripts/init/init.e3x
-r--r--r--154/50     1015 Jul 16 22:21 1996 dsa/scripts/init/init.edi
-r--r--r--154/50      732 Jul 17 23:42 1996 dsa/scripts/init/init.general
-r--r--r--154/50     1541 Jul 16 22:21 1996 dsa/scripts/init/init.isocor
-r--r--r--154/50    18348 Jul 28 22:48 1996 dsa/scripts/init/init.mhs
-r--r--r--154/50     1272 Jul 16 22:21 1996 dsa/scripts/init/init.mosaic
-r--r--r--154/50     5048 Jul 16 22:21 1996 dsa/scripts/init/init.nadf
-r--r--r--154/50     1422 Jul 16 22:21 1996 dsa/scripts/init/init.pp
-r--r--r--154/50     7660 Jul 16 22:21 1996 dsa/scripts/init/init.quipu
-r--r--r--154/50     2155 Jul 16 22:21 1996 dsa/scripts/init/init.thorn
-r--r--r--154/50     1161 Jul 16 22:21 1996 dsa/scripts/init/init.umich
-r--r--r--154/50      371 Jul 16 22:21 1996 dsa/scripts/init/init.dsp
```

```
Sep 26 15 56 1997  listing 3 Page 11

-r--r--r--154/50      470 Jul 16 22 21 1996 dsa/scripts/init/init schema
-rw-rw-rw-154/50     2921 Aug 27 17 30 1996 dsa/scripts/init/ make state
-rwxrwxr-x154/50        0 Aug 28 11 37 1996 dsa/scripts/test
-r--r--r--154/50      284 Jul 16 22 21 1996 dsa/scripts/test/Makefile
-r--r--r -154/50     1243 Jul 16 22 21 1996 dsa/scripts/test/test0
-r--r--r--154/50     9960 Jul 16 22 21 1996 dsa/scripts/test/test1
-r--r--r--154/50     8342 Jul 16 22 21 1996 dsa/scripts/test/test2
-r--r--r--154/50     9212 Jul 16 22 21 1996 dsa/scripts/test/test3
-r--r--r--154/50    11189 Jul 30 20 42 1996 dsa/scripts/test/test4
-r--r--r--154/50     3678 Jul 16 22 21 1996 dsa/scripts/test/test5
-r--r--r--154/50    11832 Jul 16 22 21 1996 dsa/scripts/test/test6
-r--r--r--154/50     5801 Jul 16 22 21 1996 dsa/scripts/test/dsptest1
-r--r--r--154/50     9971 Jul 18 00 40 1996 dsa/scripts/test/test7
-r--r--r--154/50     4168 Jul 18 00 40 1996 dsa/scripts/test/test8
-rw-rw-rw-154/50     1811 Aug 27 17 30 1996 dsa/scripts/test/ make state
-rwxrwxr-x154/50        0 Aug 28 11 37 1996 dsa/scripts/demo/
lrwxrwxrwx154/50       18 Aug 16 08 53 1996 dsa/scripts/demo/dsa symbolic link to    bin dsa rfc1006
lrwxrwxrwx154/50       15 Aug 16 08 53 1996 dsa/scripts/demo/init symbolic link to    scripts init
-r--r--r--154/50      744 Jul 16 22 21 1996 dsa/scripts/demo/Makefile
-r--r--r--154/50      581 Jul 16 22 21 1996 dsa/scripts/demo/addmgr
-r--r--r--154/50      388 Jul 16 22 21 1996 dsa/scripts/demo/demo
-r--r--r--154/50      763 Jul 16 22 21 1996 dsa/scripts/demo/demo spec
-r--r--r--154/50      628 Jul 16 22 21 1996 dsa/scripts/demo/run-dsa
-rw-rw-r--154/50   148852 Aug 16 09 49 1996 dsa/scripts/demo/demo raw
-rw-rw-rw-154/50     1607 Aug 27 17 30 1996 dsa/scripts/demo/ make state
-rwxrwxr-x154/50        0 Aug  5 05 01 1996 dsa/scripts/dua/
lrwxrwxrwx154/50       18 Aug 16 08 53 1996 dsa/scripts/dua/dua symbolic link to    bin dua rfc1006
-r--r--r--154/50      238 Apr 29 05 14 1996 dsa/scripts/dua/Makefile
-r--r--r--154/50      402 Feb 27 01 47 1996 dsa/scripts/dua/bind
-r--r--r--154/50      697 Apr 29 05 14 1996 dsa/scripts/dua/init
-rwxrwxr-x154/50        0 Aug 28 11 37 1996 dsa/scripts/data/
-r--r--r--154/50      441 Jul 16 22 21 1996 dsa/scripts/data/Makefile
-r--r--r--154/50     2531 Jul 16 22 21 1996 dsa/scripts/data/addr1
-r--r--r--154/50     4933 Jul 16 22 21 1996 dsa/scripts/data/addr2
-r--r--r--154/50      490 Jul 16 22 21 1996 dsa/scripts/data/addr3
-r--r--r--154/50    10880 Jul 16 22 21 1996 dsa/scripts/data/location
-r--r--r--154/50    11999 Jul 16 22 21 1996 dsa/scripts/data/names1
-r--r--r--154/50     4465 Jul 16 22 21 1996 dsa/scripts/data/names1x
-r--r--r--154/50    73371 Jul 16 22 21 1996 dsa/scripts/data/names2
-r--r--r--154/50     9348 Jul 16 22 21 1996 dsa/scripts/data/names2x
-r--r--r--154/50     6201 Jul 16 22 21 1996 dsa/scripts/data/orgBank
-r--r--r--154/50      610 Jul 16 22 21 1996 dsa/scripts/data/orgBankAd
-r--r--r--154/50    53178 Jul 16 22 21 1996 dsa/scripts/data/orgBrew
-r--r--r--154/50     3552 Jul 16 22 21 1996 dsa/scripts/data/orgChem
-r--r--r--154/50    55891 Jul 16 22 21 1996 dsa/scripts/data/orgTech
-r--r--r--154/50    17058 Jul 16 22 21 1996 dsa/scripts/data/orgTour
-r--r--r--154/50    24462 Jul 16 22 21 1996 dsa/scripts/data/orgTrans
-r--r--r--154/50    44711 Jul 16 22 21 1996 dsa/scripts/data/organization
-r--r--r--154/50     2742 Jul 16 22 21 1996 dsa/scripts/data/phone1
-r--r--r--154/50    20775 Jul 16 22 21 1996 dsa/scripts/data/phone2
-r--r--r--154/50     1155 Jul 16 22 21 1996 dsa/scripts/data/title1
-r--r--r--154/50     1146 Jul 16 22 21 1996 dsa/scripts/data/title1x
-r--r--r--154/50     1150 Jul 16 22 21 1996 dsa/scripts/data/title2
-r--r--r--154/50      750 Jul 16 22 21 1996 dsa/scripts/data/title2x
-r--r--r--154/50      502 Jul 16 22 21 1996 dsa/scripts/data/zfreq
-r-xr-xr-x154/50     2303 Jul 16 22 21 1996 dsa/scripts/data/zmixmerge
```

```
Sep 26 15 56 1997  listing 3 Page 12

-r--r--r--154/50      541 Jul 16 22 21 1996 dsa/scripts/data/zrev c
-r--r--r--154/50      657 Jul 16 22 21 1996 dsa/scripts/data/zstats
-rw-rw-rw-154/50     4110 Aug 27 17 30 1996 dsa/scripts/data/ make state
-rwxrwxr-x154/50        0 Aug 28 11 37 1996 dsa/scripts/orgs
-r--r--r--154/50      456 Jul 16 22 21 1996 dsa/scripts/orgs/Makefile
-r--r--r--154/50      605 Jul 16 22 21 1996 dsa/scripts/orgs/init1
-r--r--r--154/50      670 Jul 16 22 21 1996 dsa/scripts/orgs/init10
-r--r--r--154/50      323 Jul 16 22 21 1996 dsa/scripts/orgs/init100
-r--r--r--154/50      732 Jul 16 22 21 1996 dsa/scripts/orgs/init20
-r--r--r--154/50      985 Jul 16 22 21 1996 dsa/scripts/orgs/init200
-r--r--r--154/50      858 Jul 16 22 21 1996 dsa/scripts/orgs/init50
-r--r--r--154/50      496 Jul 16 22 21 1996 dsa/scripts/orgs/makeOrg1
-r--r--r--154/50      565 Jul 16 22 21 1996 dsa/scripts/orgs/makeOrg10
-r--r--r--154/50      815 Jul 16 22 21 1996 dsa/scripts/orgs/makeOrg100
-r--r--r--154/50      624 Jul 16 22 21 1996 dsa/scripts/orgs/makeOrg20
-r--r--r--154/50      875 Jul 16 22 21 1996 dsa/scripts/orgs/makeOrg200
-r--r--r--154/50      745 Jul 16 22 21 1996 dsa/scripts/orgs/makeOrg50
-r--r--r--154/50     2008 Jul 16 22 21 1996 dsa/scripts/orgs/makePop
-r--r--r--154/50      717 Jul 16 22 21 1996 dsa/scripts/orgs/ozBank spec
-r--r--r--154/50      725 Jul 16 22 21 1996 dsa/scripts/orgs/ozBankAd spec
-r--r--r--154/50      780 Jul 16 22 21 1996 dsa/scripts/orgs/ozBrew spec
-r--r--r--154/50      654 Jul 16 22 21 1996 dsa/scripts/orgs/ozChem spec
-r--r--r--154/50      718 Jul 16 22 21 1996 dsa/scripts/orgs/ozCorp spec
-r--r--r--154/50      780 Jul 16 22 21 1996 dsa/scripts/orgs/ozTech spec
-r--r--r--154/50      717 Jul 16 22 21 1996 dsa/scripts/orgs/ozTour spec
-r--r--r--154/50      722 Jul 16 22 21 1996 dsa/scripts/orgs/ozTrans spec
-rw-rw-rw-154/50     3932 Aug 27 17 30 1996 dsa/scripts/orgs/ make state
-rwxrwxr-x154/50        0 Aug 28 11 37 1996 dsa/scripts/timings
lrwxrwxrwx154/50       13 Aug 16 08 53 1996 dsa/scripts/timings/init1 symbolic link to    orgs init1
lrwxrwxrwx154/50       14 Aug 16 08 53 1996 dsa/scripts/timings/init10 symbolic link to    orgs init10
lrwxrwxrwx154/50       15 Aug 16 08 53 1996 dsa/scripts/timings/init100 symbolic link to    orgs init100
lrwxrwxrwx154/50       14 Aug 16 08 53 1996 dsa/scripts/timings/init20 symbolic link to    orgs init20
lrwxrwxrwx154/50       15 Aug 16 08 53 1996 dsa/scripts/timings/init200 symbolic link to    orgs init200
lrwxrwxrwx154/50       14 Aug 16 08 53 1996 dsa/scripts/timings/init50 symbolic link to    orgs init50
-r--r--r--154/50      494 Jul 16 22 21 1996 dsa/scripts/timings/Makefile
-r--r--r--154/50     3886 Jul 16 22 21 1996 dsa/scripts/timings/time100a
-r--r--r--154/50     3919 Jul 16 22 21 1996 dsa/scripts/timings/time100b
-r--r--r--154/50      980 Jul 16 22 21 1996 dsa/scripts/timings/time100c
-r--r--r--154/50     1385 Jul 16 22 21 1996 dsa/scripts/timings/time10a
-r--r--r--154/50     4857 Jul 16 22 21 1996 dsa/scripts/timings/time1a
-r--r--r--154/50      627 Jul 16 22 21 1996 dsa/scripts/timings/time1b
-r--r--r--154/50     2664 Jul 16 22 21 1996 dsa/scripts/timings/time1c
-r--r--r--154/50     1116 Jul 16 22 21 1996 dsa/scripts/timings/time1d
-r--r--r--154/50     2809 Jul 16 22 21 1996 dsa/scripts/timings/time1e
-r--r--r--154/50     5189 Jul 16 22 21 1996 dsa/scripts/timings/time200a
-r--r--r--154/50     5107 Jul 16 22 21 1996 dsa/scripts/timings/time200b
-r--r--r--154/50      990 Jul 16 22 21 1996 dsa/scripts/timings/time200c
-r--r--r--154/50     1137 Jul 16 22 21 1996 dsa/scripts/timings/time20a
-r--r--r--154/50     1013 Jul 16 22 21 1996 dsa/scripts/timings/time20b
-r--r--r--154/50      953 Jul 16 22 21 1996 dsa/scripts/timings/time20c
-r--r--r--154/50     2661 Jul 16 22 21 1996 dsa/scripts/timings/time50a
-r--r--r--154/50     2683 Jul 16 22 21 1996 dsa/scripts/timings/time50b
-r--r--r--154/50      769 Jul 16 22 21 1996 dsa/scripts/timings/time50c
-r--r--r--154/50      737 Jul 16 22 21 1996 dsa/scripts/timings/wtime100K
-r--r--r--154/50      663 Jul 16 22 21 1996 dsa/scripts/timings/wtime10K
-r--r--r--154/50      653 Jul 16 22 21 1996 dsa/scripts/timings/wtime1K
```

```
-r--r--r--154/50     747 Jul 16 22:21 1996 dsa/scripts/timings/wtime200K
-r--r--r--154/50     709 Jul 16 22:21 1996 dsa/scripts/timings/wtime20K
-r--r--r--154/50     725 Jul 16 22:21 1996 dsa/scripts/timings/wtime50K
-rw-rw-rw-154/50    5194 Aug 27 17:30 1996 dsa/scripts/timings/.make.state
-rwxrwxr-x154/50       0 Aug 28 11:37 1996 dsa/scripts/aliases/
-r--r--r--154/50     354 Jul 16 22:21 1996 dsa/scripts/aliases/Makefile
-r--r--r--154/50    2623 Jul 16 22:21 1996 dsa/scripts/aliases/README
-r--r--r--154/50    7952 Jul 16 22:21 1996 dsa/scripts/aliases/alias.reset
-r--r--r--154/50   10558 Jul 16 22:21 1996 dsa/scripts/aliases/alias.setup
-r--r--r--154/50    1681 Jul 16 22:21 1996 dsa/scripts/aliases/aliasreset1
-r--r--r--154/50    3682 Jul 16 22:21 1996 dsa/scripts/aliases/aliasreset4
-r--r--r--154/50   10075 Jul 30 20:42 1996 dsa/scripts/aliases/aliastest1
-r--r--r--154/50    7201 Jul 18 00:41 1996 dsa/scripts/aliases/aliastest2
-r--r--r--154/50    7695 Jul 18 00:41 1996 dsa/scripts/aliases/aliastest3
-r--r--r--154/50    7322 Jul 16 22:21 1996 dsa/scripts/aliases/aliastest4
-r--r--r--154/50    4677 Jul 18 00:41 1996 dsa/scripts/aliases/aliastest5
-r--r--r--154/50    3673 Jul 18 00:41 1996 dsa/scripts/aliases/aliastest6
-rw-rw-rw-154/50    2370 Aug 27 17:30 1996 dsa/scripts/aliases/.make.state
lrwxrwxrwx154/50      12 Aug 16 08:53 1996 dsa/scripts/bin symbolic link to ../dx500/bin
lrwxrwxrwx154/50       4 Aug 16 08:53 1996 dsa/scripts/scripts symbolic link to init
lrwxrwxrwx154/50      14 Aug 16 08:53 1996 dsa/scripts/tools symbolic link to ../dx500/tools
-r--r--r--154/50     325 Jul 16 22:21 1996 dsa/scripts/Makefile
-r--r--r--154/50    2534 Feb 26 01:48 1996 dsa/scripts/BETA.AGREEMENT
-rw-rw-rw-154/50    2482 Aug 28 11:37 1996 dsa/scripts/.make.state
-rw-rw-r--154/50    1286 Aug 28 11:37 1996 dsa/scripts/.nse_depinfo
lrwxrwxrwx154/50       8 Aug 16 08:53 1996 dsa/dx500 symbolic link to ../dx500
lrwxrwxrwx154/50       9 Aug 16 08:53 1996 dsa/stack symbolic link to ../rstack
-rw-rw-r--154/50    1124 Aug 21 12:45 1996 dsa/Makefile
-rwxrwxr-x154/50       0 Aug 20 14:13 1996 dsa/utils/
-rwxrwxr-x154/50       0 Aug 24 20:22 1996 dsa/utils/ingres/
-r--r--r--154/50     374 Jul 16 22:20 1996 dsa/utils/ingres/Makefile
-r--r--r--154/50   16008 Jul 16 22:20 1996 dsa/utils/ingres/setupDB.sc
-rw-rw-rw-154/50     837 Aug 24 20:22 1996 dsa/utils/ingres/.make.state
-rw-rw-r--154/50   23817 Aug 21 15:22 1996 dsa/utils/ingres/setupDB.c
-rw-rw-r--154/50   27380 Aug 21 15:22 1996 dsa/utils/ingres/setupDB.o
-rw-rw-r--154/50     341 Aug 21 15:22 1996 dsa/utils/ingres/.nse_depinfo
-rw-rw-r--154/50     868 Aug 24 20:25 1996 dsa/utils/Makefile
-rwxrwxr-x154/50       0 Aug 28 11:37 1996 dsa/utils/tools/
-r--r--r--154/50     644 Jul 16 22:21 1996 dsa/utils/tools/Makefile
-r--r--r--154/50     895 Jul 16 22:20 1996 dsa/utils/tools/dxdelete
-r--r--r--154/50    3574 Jul 16 22:20 1996 dsa/utils/tools/dxexport
-r--r--r--154/50    3283 Jul 16 22:20 1996 dsa/utils/tools/dxgetnodes
-r--r--r--154/50    8072 Jul 16 22:20 1996 dsa/utils/tools/dxhelp
-r--r--r--154/50    4242 Jul 16 22:20 1996 dsa/utils/tools/dximport
-r--r--r--154/50     974 Jul 16 22:20 1996 dsa/utils/tools/dxnewdb
-r--r--r--154/50    3019 Jul 16 22:20 1996 dsa/utils/tools/dxprepare
-r--r--r--154/50    1208 Jul 16 22:20 1996 dsa/utils/tools/dxprintfield
-r--r--r--154/50     881 Jul 16 22:20 1996 dsa/utils/tools/dxreload
-r--r--r--154/50    1606 Jul 16 22:20 1996 dsa/utils/tools/dxtranslate
-r--r--r--154/50    1710 Jul 16 22:20 1996 dsa/utils/tools/dxtunedb
-r--r--r--154/50   11470 Jul 16 22:20 1996 dsa/utils/tools/dxupdate
-r--r--r--154/50     680 Jul 16 22:20 1996 dsa/utils/tools/xaddfields.awk
-r--r--r--154/50     678 Jul 16 22:20 1996 dsa/utils/tools/xaddquotes.awk
-r--r--r--154/50    5822 Jul 16 22:20 1996 dsa/utils/tools/xcheckspec.awk
-r--r--r--154/50     960 Jul 16 22:20 1996 dsa/utils/tools/xchoplast.awk
-r--r--r--154/50    1404 Jul 16 22:20 1996 dsa/utils/tools/xcodes2awk.awk
```

```
-r--r--r--154/50    4541 Jul 16 22:20 1996 dsa/utils/tools/xdat2ddx.awk
-r--r--r--154/50    5517 Jul 16 22:20 1996 dsa/utils/tools/xdat2mod.awk
-r--r--r--154/50    5522 Jul 16 22:20 1996 dsa/utils/tools/xddx2dat.awk
-r--r--r--154/50    1341 Jul 16 22:20 1996 dsa/utils/tools/xdelete.awk
-r--r--r--154/50     277 Jul 16 22:20 1996 dsa/utils/tools/xgetdit.awk
-r--r--r--154/50    5461 Jul 16 22:20 1996 dsa/utils/tools/xddx2del.awk
-r--r--r--154/50    6836 Jul 16 22:20 1996 dsa/utils/tools/xddx2pop.awk
-r--r--r--154/50    7882 Jul 16 22:20 1996 dsa/utils/tools/xlog2ddx.awk
-r--r--r--154/50    4132 Jul 16 22:20 1996 dsa/utils/tools/xmerge.awk
-r--r--r--154/50    8122 Jul 16 22.21 1996 dsa/utils/tools/xmod2pop.awk
-r--r--r--154/50     524 Jul 16 22:21 1996 dsa/utils/tools/xprefixddx.awk
-r--r--r--154/50    2360 Jul 16 22:21 1996 dsa/utils/tools/xreload.awk
-r--r--r--154/50     909 Jul 16 22:21 1996 dsa/utils/tools/xtpt2comma.awk
-r--r--r--154/50     409 Jul 16 22:21 1996 dsa/utils/tools/xunformat.awk
-r--r--r--154/50    6786 Jul 16 22:21 1996 dsa/utils/tools/xupdate.awk
-r--r--r--154/50    3013 Jul 16 22:20 1996 dsa/utils/tools/dxbackup
-rw-rw-rw-154/50    5572 Aug 27 17:30 1996 dsa/utils/tools/.make.state
lrwxrwxrwx154/50       8 Aug 16 08:53 1996 dsa/utils/dx500 symbolic link to ../dx500
-rwxrwxr-x154/50       0 Aug 28 11:37 1996 dsa/utils/snmp/
lrwxrwxrwx154/50      38 Aug 16 08:53 1996 dsa/utils/snmp/dxmonitor symbolic link to ../../../dxmonitor/dxmonitor/dxmonitor
-r--r--r--154/50     450 Aug  5 04:42 1996 dsa/utils/snmp/Makefile
-r--r--r--154/50     910 Jul 16 22:21 1996 dsa/utils/snmp/README
-r--r--r--154/50    5870 Jul 16 22:21 1996 dsa/utils/snmp/asn1dec.c
-r--r--r--154/50     566 Jul 16 22.21 1996 dsa/utils/snmp/dxsnmp
-r--r--r--154/50    1443 Jul 16 22:21 1996 dsa/utils/snmp/mib.h
-r--r--r--154/50    4492 Jul 16 22:21 1996 dsa/utils/snmp/snmplib.h
-r--r--r--154/50    4669 Jul 16 22:21 1996 dsa/utils/snmp/walker.c
-rw-rw-rw-154/50    1761 Aug 27 18:05 1996 dsa/utils/snmp/.make.state
-rw-rw-r--154/50   14516 Aug 27 18:05 1996 dsa/utils/snmp/walker.o
-rw-rw-r--154/50   10124 Aug 27 18:05 1996 dsa/utils/snmp/asn1dec.o
-rw-rw-r--154/50     377 Aug 27 18:05 1996 dsa/utils/snmp/.nse_depinfo
-rwxrwxr-x154/50       0 Aug 28 11:37 1996 dsa/utils/cmip/
lrwxrwxrwx154/50      26 Aug 16 08:53 1996 dsa/utils/cmip/Htcp.c symbolic link to ../../stack/network/Htcp.c
-r--r--r--154/50     495 Jul 18 23:44 1996 dsa/utils/cmip/Makefile
-r--r--r--154/50     508 Jul 18 23:44 1996 dsa/utils/cmip/README
-r--r--r--154/50   16487 Jul 18 00:38 1996 dsa/utils/cmip/cmipmgr.c
-r--r--r--154/50   19672 Jul 18 00:38 1996 dsa/utils/cmip/dsa_mgr_mib.c
-r--r--r--154/50     644 Jul 16 22:21 1996 dsa/utils/cmip/dxcmip
-r--r--r--154/50     627 Jul 18 00.38 1996 dsa/utils/cmip/mfix.c
-rw-rw-rw-154/50    3001 Aug 27 18:05 1996 dsa/utils/cmip/.make.state
-rw-rw-r--154/50   41924 Aug 27 18:05 1996 dsa/utils/cmip/cmipmgr.o
-rw-rw-r--154/50   51184 Aug 27 18:05 1996 dsa/utils/cmip/dsa_mgr_mib.o
-rw-rw-r--154/50     558 Aug 27 18:05 1996 dsa/utils/cmip/.nse_depinfo
-rw-rw-r--154/50    2788 Aug 27 18:05 1996 dsa/utils/cmip/mfix.o
-rw-rw-r--154/50   31920 Aug 27 18:05 1996 dsa/utils/cmip/Htcp.o
lrwxrwxrwx154/50       9 Aug 16 08:53 1996 dsa/utils/bin symbolic link to dx500/bin
-rwxrwxr-x154/50       0 Aug 28 11:37 1996 dsa/utils/disp/
lrwxrwxrwx154/50      18 Aug 16 08:53 1996 dsa/utils/disp/dispCopy symbolic link to ../../bin/dispCopy
-r--r--r--154/50   16093 Jul 16 22:21 1996 dsa/utils/disp/Htcp.c
-r--r--r--154/50     523 Jul 16 22:21 1996 dsa/utils/disp/Makefile
-r--r--r--154/50     495 Jul 16 22:21 1996 dsa/utils/disp/README
-r--r--r--154/50     643 Jul 16 22:21 1996 dsa/utils/disp/dbif.h
-r--r--r--154/50    7407 Jul 18 23:44 1996 dsa/utils/disp/dbif_dap.c
-r--r--r--154/50    8870 Jul 16 22:21 1996 dsa/utils/disp/dbif_test.c
-r--r--r--154/50     365 Jul 16 22:21 1996 dsa/utils/disp/dispCopy.cfg
-r--r--r--154/50    6390 Jul 16 22:21 1996 dsa/utils/disp/master.c
```

```
-r--r--r--154/50    8174 Jul 16 22 21 1996 dsa/utils/disp/mconfig c
-r--r--r--154/50    8428 Jul 16 22 21 1996 dsa/utils/disp/mupdate c
-r--r--r--154/50    1864 Jul 16 22 21 1996 dsa/utils/disp shadow h
-rw-rw-rw-154/50    4779 Aug 27 18 04 1996 dsa/utils/disp/ make state
-rw-rw-r--154/50   41728 Aug 27 18 04 1996 dsa/utils/disp/dbif_dap o
-rw-rw-r--154/50   49656 Aug 27 18 04 1996 dsa/utils/disp/master o
-rw-rw-r--154/50     631 Aug 27 18 04 1996 dsa/utils/disp/ nse_depinfo
-rw-rw-r--154/50   52144 Aug 27 18 04 1996 dsa/utils/disp mconfig o
-rw-rw-r--154/50   49400 Aug 27 18 04 1996 dsa/utils/disp/mupdate o
-rw-rw-r--154/50   31920 Aug 27 18 04 1996 dsa/utils/disp/Htcp o
-rwxrwxr-x154/50       0 Aug 28 11 37 1996 dsa/utils/dua/
lrwxrwxrwx154/50      21 Aug 16 08 53 1996 dsa/utils/dua/dua symbolic link to       bin dua rf-1006
-r--r--r--154/50     243 Jul 16 22 21 1996 dsa/utils/dua/Makefile
-r--r--r--154/50     679 Jul 16 22:21 1996 dsa/utils dua init
-rw-rw-rw-154/50     868 Aug 27 17 10 1996 dsa/utils/dua/ make state
-rwxrwxr-x154/50       0 Aug 28 11 37 1996 dsa/utils/actest/
lrwxrwxrwx154/50      24 Aug 16 08 53 1996 dsa/utils/actest/BasicAC h symbolic link to       stack ac BasicAC h
lrwxrwxrwx154/50      28 Aug 16 08 53 1996 dsa/utils/actest README symbolic link to      /stack ac actest readme
lrwxrwxrwx154/50      19 Aug 16 08 53 1996 dsa/utils/actest/ac c symbolic link to   /  stack/ac ac c
lrwxrwxrwx154/50      19 Aug 16 08 53 1996 dsa/utils/actest/ac h symbolic link to   /  /stack/ac ac h
lrwxrwxrwx154/50      23 Aug 16 08 53 1996 dsa/utils/actest/acdump c symbolic link to       stack/ac acdump c
lrwxrwxrwx154/50      16 Aug 16 08 53 1996 dsa/utils/actest actest symbolic link to   /  bin actest
lrwxrwxrwx154/50      23 Aug 16 08 53 1996 dsa/utils/actest/actest c symbolic link to   /   stack ac actest c
lrwxrwxrwx154/50      24 Aug 16 08 53 1996 dsa/utils/actest/actest in symbolic link to       stack ac actest in
lrwxrwxrwx154/50      26 Aug 16 08 53 1996 dsa/utils/actest/basicAC asn symbolic link to      /stack ac basicAC asn
-r--r--r--154/50     461 Aug  1 18 56 1996 dsa/utils/actest Makefile
lrwxrwxrwx154/50      22 Aug 16 08 53 1996 dsa/utils/actest acadd c symbolic link to       stack ac acadd c
-rw-rw-rw-154/50    2979 Aug 27 18 06 1996 dsa/utils/actest  make state
-rw-rw-r--154/50   71092 Aug 27 18 05 1996 dsa/utils/actest/actest o
-rw-rw-r--154/50   87068 Aug 27 18 05 1996 dsa/utils/actest/ac o
-rw-rw-r--154/50     549 Aug 27 18 06 1996 dsa/utils/actest  nse_depinfo
-rw-rw-r--154/50   68156 Aug 27 18 05 1996 dsa/utils/actest acadd o
-rw-rw-r--154/50   46044 Aug 27 18 06 1996 dsa/utils/actest acdump o
-rwxrwxr-x154/50       0 Aug 28 11 37 1996 dsa/utils/rubix/
-rw-rw-r--154/50     394 Aug 26 14 54 1996 dsa/utils/rubix/Makefile
-rw-rw-rw-154/50    1315 Aug 27 18 04 1996 dsa/utils/rubix/ make state
-rw-rw-r--154/50   10623 Aug 27 09 21 1996 dsa/utils/rubix/setupDB e
-rw-rw-r--154/50   18954 Aug 27 18 04 1996 dsa/utils/rubix/setupDB c
-rw-rw-r--154/50   18954 Aug 27 18 04 1996 dsa utils/rubix/setupDB pc
lrwxrwxrwx154/50      17 Aug 22 11 27 1996 dsa utils/rubix/z symbolic link to       bin setupDB
-rw-rw-r--154/50     237 Aug 26 14 55 1996 dsa/utils/rubix drop sql
-rw-rw-r--154/50     342 Aug 27 18 04 1996 dsa/utils/rubix/ nse_depinfo
lrwxrwxrwx154/50      25 Aug 26 14:52 1996 dsa/utils rubix/where awk symbolic link to       dip rubix where awk
-rw-rw-r--154/50     129 Aug 26 20 20 1996 dsa/utils/rubix/create_dit sql
-rw-rw-r--154/50   33284 Aug 27 18 04 1996 dsa/utils rubix/setupDB o
-rw-rw-r--154/50     183 Aug 27 14 33 1996 dsa/utils/rubix/delete sql
-rwxrwxr-x154/50       0 Aug 28 15 14 1996 dsa/lib/
-rw-rw-r--154/50  323196 Aug 28 15 14 1996 dsa/lib oper a
-rw-rw-r--154/50  497330 Aug 27 17 42 1996 dsa lib/schema a
-rw-rw-r--154/50  366436 Aug 27 17 43 1996 dsa/lib/access a
-rw-rw-r--154/50 1618942 Aug 28 15 14 1996 dsa/lib/mgmt a
-rw-rw-r--154/50  680014 Aug 27 17 47 1996 dsa lib/support a
-rw-rw-r--154/50 1693852 Aug 28 15 15 1996 dsa/lib/dip a
-r--r--r--154/50    8602 Aug  5 04 07 1996 dsa ChangeLog
-r--r--r--154/50    3149 Jul 31 06 25 1996 dsa DX500-V3 0
-rwxrwxr-x154/50       0 Sep 17 17 14 1996 dsa access/
```

```
lrwxrwxrwx154/50      21 Aug 16 08 53 1996 dsa access/BasicAC h symbolic link to    stack ac BasicAC h
lrwxrwxrwx154/50      16 Aug 16 08 53 1996 dsa/access/ac c symbolic link to    stack ac ac c
lrwxrwxrwx154/50      16 Aug 16 08 53 1996 dsa/access ac h symbolic link to    stack ac ac h
lrwxrwxrwx154/50      20 Aug 16 08 53 1996 dsa/access/acdump c symbolic link to    stack ac acdump c
lrwxrwxrwx154/50      23 Aug 16 08 53 1996 dsa/access basicAC asn symbolic link to    stack ac basicAC asn
-r--r--r--154/50     327 Aug  1 18 54 1996 dsa/access/Makefile
-r--r--r--154/50   26408 Aug  5 03 30 1996 dsa/access/acMain c
-r--r--r--154/50   23650 Aug  5 03 30 1996 dsa/access acRules c
lrwxrwxrwx154/50      19 Aug 16 08 53 1996 dsa/access acadd c symbolic link to    stack ac acadd c
-r--r--r--154/50     714 Aug  5 03 30 1996 dsa/access/acDump c
-rw-rw-rw-154/50    3378 Aug 27 17 43 1996 dsa access/ make state
-rw-rw-r--154/50   78684 Aug 27 17 42 1996 dsa access acMain o
-rw-rw-r--154/50   72492 Aug 27 17 42 1996 dsa access acRules o
-rw-rw-r--154/50     401 Aug 27 17 42 1996 dsa/access  nse_depinfo
-rw-rw-r--154/50   58884 Aug 27 17 42 1996 dsa access acDump o
-rw-rw-r--154/50   87024 Aug 27 17 42 1996 dsa/access ac o
-rw-rw-r--154/50   68108 Aug 27 17 43 1996 dsa access/acadd o
lrwxrwxrwx154/50       9 Aug 16 08 53 1996 dsa/bin symbolic link to dx500 bin
-rwxrwxr-x154/50       0 Aug 27 18 06 1996 dx500
-rwxrwxr-x154/50    3423 Jun 24 01 29 1996 dx500 MAKETAPE
-rwxrwxr-x154/50       0 Aug  5 05 11 1996 dx500 test
-rwxrwxr-x154/50       0 Aug 27 18 06 1996 dx500 test/aliases
-r--r--r--154/50    2623 Jul 16 22 21 1996 dx500 test/aliases README
-r--r--r--154/50    7952 Jul 16 22 21 1996 dx500 test aliases alias reset
-r--r--r--154/50   10558 Jul 16 22 21 1996 dx500 test aliases alias setup
-r--r--r--154/50    1681 Jul 16 22 21 1996 dx500 test/aliases aliasreset1
-r--r--r--154/50    3682 Jul 16 22 21 1996 dx500 test aliases aliasreset4
-r--r--r--154/50   10075 Jul 30 20 42 1996 dx500 test aliases aliastest1
-r--r--r--154/50    7201 Jul 18 00 41 1996 dx500 test/aliases aliastest2
-r--r--r--154/50    7695 Jul 18 00 41 1996 dx500 test aliases aliastest3
-r--r--r--154/50    7322 Jul 16 22 21 1996 dx500 test aliases aliastest4
-r--r--r--154/50    4677 Jul 18 00 41 1996 dx500 test aliases aliastest5
-r--r--r--154/50    3673 Jul 18 00 41 1996 dx500 test aliases aliastest6
-rwxrwxr-x154/50       0 Aug 27 18 06 1996 dx500 test data/
-r-xr-xr-x154/50     502 Jul 16 22 21 1996 dx500 test data zfreq
-r-xr-xr-x154/50    2303 Jul 16 22 21 1996 dx500 test/data zmixmerge
-r-xr-xr-x154/50     657 Jul 16 22 21 1996 dx500 test data zstats
-r--r--r--154/50    2531 Jul 16 22 21 1996 dx500 test data addr1
-r--r--r--154/50    4933 Jul 16 22 21 1996 dx500 test data addr2
-r--r--r--154/50     490 Jul 16 22 21 1996 dx500 test data addr3
-r--r--r--154/50   10880 Jul 16 22 21 1996 dx500 test data location
-r--r--r--154/50   11999 Jul 16 22 21 1996 dx500 test/data names1
-r--r--r--154/50    4465 Jul 16 22 21 1996 dx500 test data names1x
-r--r--r--154/50   73371 Jul 16 22 21 1996 dx500 test data names2
-r--r--r--154/50    9348 Jul 16 22 21 1996 dx500 test data names2x
-r--r--r--154/50    6201 Jul 16 22 21 1996 dx500 test data orgBank
-r--r--r--154/50     610 Jul 16 22 21 1996 dx500 test data orgBankAd
-r--r--r--154/50   53178 Jul 16 22 21 1996 dx500 test data orgBrew
-r--r--r--154/50    3552 Jul 16 22 21 1996 dx500 test data orgChem
-r--r--r--154/50   55891 Jul 16 22 21 1996 dx500 test data/orgTech
-r--r--r--154/50   17058 Jul 16 22 21 1996 dx500 test data orgTour
-r--r--r--154/50   24462 Jul 16 22 21 1996 dx500 test data orgTrans
-r--r--r--154/50   44711 Jul 16 22 21 1996 dx500 test data organization
-r--r--r--154/50    2742 Jul 16 22 21 1996 dx500 test data/phone1
-r--r--r--154/50   20775 Jul 16 22 21 1996 dx500 test data phone2
-r--r--r--154/50    1155 Jul 16 22 21 1996 dx500 test data title1
```

```
-r--r--r--154/50    1146 Jul 16 22:21 1996 dx500/test/data/title1x
-r--r--r--154/50    1150 Jul 16 22:21 1996 dx500/test/data/title2
-r--r--r--154/50     750 Jul 16 22:21 1996 dx500/test/data/title2x
-rwxrwxr-x154/50       0 Aug 27 18:06 1996 dx500/test/orgs/
-r-xr-xr-x154/50     496 Jul 16 22:21 1996 dx500/test/orgs/makeOrg1
-r-xr-xr-x154/50     565 Jul 16 22:21 1996 dx500/test/orgs/makeOrg10
-r-xr-xr-x154/50     815 Jul 16 22:21 1996 dx500/test/orgs/makeOrg100
-r-xr-xr-x154/50     624 Jul 16 22:21 1996 dx500/test/orgs/makeOrg20
-r-xr-xr-x154/50     875 Jul 16 22:21 1996 dx500/test/orgs/makeOrg200
-r-xr-xr-x154/50     745 Jul 16 22:21 1996 dx500/test/orgs/makeOrg50
-r-xr-xr-x154/50    2008 Jul 16 22:21 1996 dx500/test/orgs/makePop
-r--r--r--154/50     605 Jul 16 22:21 1996 dx500/test/orgs/init1
-r--r--r--154/50     670 Jul 16 22:21 1996 dx500/test/orgs/init10
-r--r--r--154/50     923 Jul 16 22:21 1996 dx500/test/orgs/init100
-r--r--r--154/50     732 Jul 16 22:21 1996 dx500/test/orgs/init20
-r--r--r--154/50     985 Jul 16 22.21 1996 dx500/test/orgs/init200
-r--r--r--154/50     858 Jul 16 22:21 1996 dx500/test/orgs/init50
-r--r--r--154/50     717 Jul 16 22:21 1996 dx500/test/orgs/ozBank.spec
-r--r--r--154/50     725 Jul 16 22:21 1996 dx500/test/orgs/ozBankAd.spec
-r--r--r--154/50     780 Jul 16 22:21 1996 dx500/test/orgs/ozBrew.spec
-r--r--r--154/50     654 Jul 16 22:21 1996 dx500/test/orgs/ozChem.spec
-r--r--r--154/50     718 Jul 16 22:21 1996 dx500/test/orgs/ozCorp.spec
-r--r--r--154/50     780 Jul 16 22:21 1996 dx500/test/orgs/ozTech.spec
-r--r--r--154/50     717 Jul 16 22:21 1996 dx500/test/orgs/ozTour.spec
-r--r--r--154/50     722 Jul 16 22:21 1996 dx500/test/orgs/ozTrans.spec
-rwxrwxr-x154/50       0 Aug 27 18:06 1996 dx500/test/timings/
-r--r--r--154/50    3886 Jul 16 22:21 1996 dx500/test/timings/time100a
-r--r--r--154/50    3919 Jul 16 22:21 1996 dx500/test/timings/time100b
-r--r--r--154/50     980 Jul 16 22:21 1996 dx500/test/timings/time100c
-r--r--r--154/50    1385 Jul 16 22:21 1996 dx500/test/timings/time10a
-r--r--r--154/50    4857 Jul 16 22:21 1996 dx500/test/timings/time1a
-r--r--r--154/50     627 Jul 16 22:21 1996 dx500/test/timings/time1b
-r--r--r--154/50    2664 Jul 16 22:21 1996 dx500/test/timings/time1c
-r--r--r--154/50    1116 Jul 16 22:21 1996 dx500/test/timings/time1d
-r--r--r--154/50    2809 Jul 16 22:21 1996 dx500/test/timings/time1e
-r--r--r--154/50    5189 Jul 16 22:21 1996 dx500/test/timings/time200a
-r--r--r--154/50    5107 Jul 16 22:21 1996 dx500/test/timings/time200b
-r--r--r--154/50     990 Jul 16 22:21 1996 dx500/test/timings/time200c
-r--r--r--154/50    1137 Jul 16 22:21 1996 dx500/test/timings/time20a
-r--r--r--154/50    1013 Jul 16 22:21 1996 dx500/test/timings/time20b
-r--r--r--154/50     953 Jul 16 22:21 1996 dx500/test/timings/time20c
-r--r--r--154/50    2661 Jul 16 22:21 1996 dx500/test/timings/time50a
-r--r--r--154/50    2683 Jul 16 22:21 1996 dx500/test/timings/time50b
-r--r--r--154/50     969 Jul 16 22:21 1996 dx500/test/timings/time50c
-r--r--r--154/50     737 Jul 16 22:21 1996 dx500/test/timings/wtime100K
-r--r--r--154/50     663 Jul 16 22:21 1996 dx500/test/timings/wtime10K
-r--r--r--154/50     653 Jul 16 22:21 1996 dx500/test/timings/wtime1K
-r--r--r--154/50     747 Jul 16 22:21 1996 dx500/test/timings/wtime200K
-r--r--r--154/50     709 Jul 16 22:21 1996 dx500/test/timings/wtime20K
-r--r--r--154/50     725 Jul 16 22:21 1996 dx500/test/timings/wtime50K
lrwxrwxrwx154/50      13 Aug 27 18:06 1996 dx500/test/timings/init1 symbolic link to ../orgs/init1
lrwxrwxrwx154/50      14 Aug 27 18:06 1996 dx500/test/timings/init10 symbolic link to ../orgs/init10
lrwxrwxrwx154/50      15 Aug 27 18:06 1996 dx500/test/timings/init100 symbolic link to ../orgs/init100
lrwxrwxrwx154/50      14 Aug 27 18:06 1996 dx500/test/timings/init20 symbolic link to ../orgs/init20
lrwxrwxrwx154/50      15 Aug 27 18:06 1996 dx500/test/timings/init200 symbolic link to ../orgs/init200
lrwxrwxrwx154/50      14 Aug 27 18:06 1996 dx500/test/timings/init50 symbolic link to ../orgs/init50
```

```
-rwxrwxr-x154/50       0 Aug 28 14:39 1996 dx500/test/world/
-rwxrwxr-x154/50     317 Oct 20 00:59 1995 dx500/test/world/cadsIII
-rw-rw-r--154/50     799 Oct 25 06:34 1995 dx500/test/world/dcphone.spec
-rw-rw-r--154/50     980 Jul 21 17:43 1996 dx500/test/world/init
-rw-rw-r--154/50    7123 Oct 19 23:04 1995 dx500/test/world/misx.codes
-rw-r--r--154/50    3636 Apr  3 04:42 1996 dx500/test/world/init.mis
-rw-rw-r--154/50     909 Oct 25 07:24 1995 dx500/test/world/product.spec
-rwxrwxr-x154/50     418 Oct 24 20:21 1995 dx500/test/world/mis
lrwxrwxrwx154/50      18 Aug 16 08:54 1996 dx500/test/world/dsa symbolic link to ../bin/dsa.rfc1006
-rwxrwxr-x154/50     374 Jun 29 20:21 1995 dx500/test/world/army
-rwxrwxr-x154/50     317 Oct 20 00:59 1995 dx500/test/world/dcphone
-rw-r--r--154/50     554 Oct 25 21:47 1995 dx500/test/world/army.codes
-rw-rw-r--154/50    1344 Oct 25 06:35 1995 dx500/test/world/misx.spec
-rwxrwxr-x154/50     317 Oct 20 00:56 1995 dx500/test/world/product
-rw-rw-r--154/50    1736 Oct 25 06:34 1995 dx500/test/world/cadsIII.spec
-rw-r--r--154/50    2414 Apr  3 04:42 1996 dx500/test/world/init.product
-rw-rw-r--154/50  224404 Mar  3 01:27 1995 dx500/test/world/product.raw
-rw-rw-r--154/50    5458 Apr  3 04:43 1996 dx500/test/world/init.cads
-rw-rw-r--154/50    1262 Oct 25 08:02 1995 dx500/test/world/armyx.spec
-rw-rw-r--154/50    1262 Apr  3 03:15 1996 dx500/test/world/init.army
-rw-r--r--154/50    2369 Jul 10 22:41 1996 dx500/test/world/init.dcphone
-rwxrwxr-x154/50     299 May  2 23:06 1995 dx500/test/world/demo
-rw-rw-r--154/50     832 Oct 25 06:35 1995 dx500/test/world/demo.spec
-rwxrwxr-x154/50     375 Apr  3 05:45 1996 dx500/test/world/makeWorld
-rw-rw-r--154/50     573 Apr  3 05:46 1996 dx500/test/world/init2
lrwxrwxrwx154/50      14 Aug 16 08:54 1996 dx500/test/world/dsatcp symbolic link to ../bin/dsa.tcp
-rwxrwxr-x154/50   25707 Apr  3 03:28 1996 dx500/test/world/dx500dua.cfg
-rwxrwxr-x154/50     516 Apr 23 03:21 1996 dx500/test/world/init.dcsec
-rw-r--r--154/50       0 Oct 25 22:37 1995 dx500/test/DX500-TEST-DIR
lrwxrwxrwx154/50      10 Aug 16 08:54 1996 dx500/test/scripts symbolic link to  ./scripts
lrwxrwxrwx154/50       6 Aug 16 08:54 1996 dx500/test/bin symbolic link to ../bin
-rwxrwxr-x154/50       0 Aug 28 14:39 1996 dx500/demo/
-r-xr-xr-x154/50     388 Jul 16 22:21 1996 dx500/demo/demo
-r-xr-xr-x154/50     628 Jul 16 22:21 1996 dx500/demo/run-dsa
-rw-rw-r--154/50  148852 Aug 16 09:49 1996 dx500/demo/demo.raw
-r--r--r--154/50     763 Jul 16 22:21 1996 dx500/demo/demo.spec
-rw-rw-r--154/50     954 Aug 28 13:52 1996 dx500/demo/init
lrwxrwxrwx154/50      18 Aug 27 18:06 1996 dx500/demo/dsa symbolic link to ../bin/dsa.rfc1006
-rw-rw-r--154/50      33 Aug 20 18:49 1996 dx500/demo/z
-rw-rw-r--154/50      66 Aug 26 18:22 1996 dx500/demo/a
-rw-rw-r--154/50  187132 Aug 26 18:25 1996 dx500/demo/demo.dat
-rw-rw-r--154/50     978 Aug 26 19:17 1996 dx500/demo/test0
-rw-rw-r--154/50    9960 Jul 16 22:21 1996 dx500/demo/test1
-rw-rw-r--154/50    8344 Aug 27 18:35 1996 dx500/demo/test2
-rw-rw-r--154/50    6336 Aug 28 09:50 1996 dx500/demo/test
-r--r--r--154/50    9212 Jul 16 22:21 1996 dx500/demo/test3
-rw-rw-r--154/50     864 Aug 28 11:15 1996 dx500/demo/zinit
-rw-rw-r--154/50  931909 Aug 26 18:31 1996 dx500/demo/zdemo.pop
-rw-rw-r--154/50     174 Aug 26 18:31 1996 dx500/demo/demo.pop
-rw-rw-r--154/50    3708 Aug 28 13:53 1996 dx500/demo/log.rubix
-rw-rw-r--154/50      11 Aug 26 18:27 1996 dx500/demo/demo1.dat
-rw-rw-r--154/50    3692 Aug 26 18:27 1996 dx500/demo/demo2.dat
-rw-rw-r--154/50    3180 Aug 28 13:05 1996 dx500/demo/log.rubix1
-rwxrwxr-x154/50       0 Sep 17 17:14 1996 dx500/bin/
-rwxrwxr-x154/50 2236416 Sep 17 17:14 1996 dx500/bin/dsa.rfc1006
-rwxrwxr-x154/50  262144 Aug 27 18:05 1996 dx500/bin/snmpWalker
```

```
Sep 26 15.56 1997  listing 3 Page 19

-rwxrwxr-x154/50  663552 Aug 27 18 05 1996 dx500/bin cmipMgr
-rwxrwxr-x154/50  352256 Aug 27 18 06 1996 dx500/bin/actest
-rwxrwxr-x154/50       0 Aug 27 18 03 1996 dx500/tools/
-r-xr-xr-x154/50    3013 Jul 16 22 20 1996 dx500/tools/dxbackup
-r-xr-xr-x154/50     895 Jul 16 22 20 1996 dx500/tools dxdelete
-r-xr-xr-x154/50    3574 Jul 16 22 20 1996 dx500/tools/dxexport
-r-xr-xr-x154/50    3283 Jul 16 22 20 1996 dx500/tools/dxgetnodes
-r-xr-xr-x154/50    8072 Jul 16 22 20 1996 dx500 tools/dxhelp
-r-xr-xr-x154/50    4242 Jul 16 22 20 1996 dx500 tools/dximport
-r-xr-xr-x154/50     974 Jul 16 22 20 1996 dx500/tools/dxnewdb
-r-xr-xr-x154/50    3019 Jul 16 22 20 1996 dx500/tools/dxprepare
-r-xr-xr-x154/50    1208 Jul 16 22 20 1996 dx500/tools/dxprintfield
-r-xr-xr-x154/50     881 Jul 16 22 20 1996 dx500/tools/dxreload
-r-xr-xr-x154/50    1606 Jul 16 22 20 1996 dx500/tools/dxtranslate
-r-xr-xr-x154/50    1710 Jul 16 22.20 1996 dx500/tools/dxtunedb
-r-xr-xr-x154/50   11470 Jul 16 22 20 1996 dx500/tools/dxupdate
-r--r--r--154/50     680 Jul 16 22 20 1996 dx500/tools/xaddfields awk
-r--r--r--154/50     678 Jul 16 22.20 1996 dx500/tools/xaddquotes awk
-r--r--r--154/50    5822 Jul 16 22 20 1996 dx500/tools/xcheckspec awk
-r--r--r--154/50     960 Jul 16 22:20 1996 dx500/tools/xchoplast awk
-r--r--r--154/50    1404 Jul 16 22 20 1996 dx500/tools/xcodes2awk awk
-r--r--r--154/50    4541 Jul 16 22 20 1996 dx500/tools/xdat2ddx awk
-r--r--r--154/50    5517 Jul 16 22:20 1996 dx500/tools/xdat2mod awk
-r--r--r--154/50    5522 Jul 16 22 20 1996 dx500/tools/xddx2dat awk
-r--r--r--154/50    5461 Jul 16 22 20 1996 dx500/tools/xddx2del awk
-r--r--r--154/50    6836 Jul 16 22.20 1996 dx500/tools/xddx2pop awk
-r--r--r--154/50    1341 Jul 16 22 20 1996 dx500/tools/xdelete awk
-r--r--r--154/50     277 Jul 16 22 20 1996 dx500/tools/xgetdit awk
-r--r--r--154/50    7882 Jul 16 22 20 1996 dx500/tools/xlog2ddx awk
-r--r--r--154/50    4132 Jul 16 22 20 1996 dx500/tools/xmerge awk
-r--r--r--154/50    8122 Jul 16 22 21 1996 dx500/tools/xmod2pop awk
-r--r--r--154/50     524 Jul 16 22 21 1996 dx500/tools/xprefixddx awk
-r--r--r--154/50    2360 Jul 16 22.21 1996 dx500/tools/xreload awk
-r--r--r--154/50     909 Jul 16 22.21 1996 dx500/tools/xtpt2comma awk
-r--r--r--154/50     409 Jul 16 22 21 1996 dx500/tools/xunformat awk
-r--r--r--154/50    6786 Jul 16 22.21 1996 dx500/tools/xupdate awk
-rwxrwxr-x154/50       0 Aug 27 18 06 1996 dx500/scripts/
-r--r--r--154/50     864 Jul 31 05 43 1996 dx500/scripts/init
-r--r--r--154/50   14902 Jul 16 22 21 1996 dx500/scripts/init attr
-r--r--r--154/50   14440 Jul 16 22 21 1996 dx500/scripts/init cosine
-r--r--r--154/50   11283 Jul 16 22 21 1996 dx500/scripts/init dms
-r--r--r--154/50     371 Jul 16 22 21 1996 dx500/scripts/init dsp
-r--r--r--154/50    4948 Jul 16 22 21 1996 dx500/scripts/init e3x
-r--r--r--154/50    1015 Jul 16 22 21 1996 dx500/scripts/init edi
-r--r--r--154/50     732 Jul 17 23 42 1996 dx500/scripts/init general
-r--r--r--154/50    1541 Jul 16 22 21 1996 dx500/scripts/init isocor
-r--r--r--154/50   18348 Jul 28 22 48 1996 dx500/scripts/init mhs
-r--r--r--154/50    1272 Jul 16 22 21 1996 dx500/scripts/init mosaic
-r--r--r--154/50    5048 Jul 16 22 21 1996 dx500/scripts/init nadf
-r--r--r--154/50    1422 Jul 16 22.21 1996 dx500 scripts/init pp
-r--r--r--154/50    7660 Jul 16 22 21 1996 dx500 scripts/init quipu
-r--r--r--154/50     470 Jul 16 22 21 1996 dx500/scripts/init schema
-r--r--r--154/50    2155 Jul 16 22.21 1996 dx500/scripts/init thorn
-r--r--r--154/50    1161 Jul 16 22 21 1996 dx500 scripts/init umich
-r--r--r--154/50    1243 Jul 16 22 21 1996 dx500/scripts/test0
-r--r--r--154/50    9960 Jul 16 22 21 1996 dx500 scripts/test1
```

```
Sep 26 15:56 1997  listing 3 Page 20

-r--r--r--154/50    8342 Jul 16 22 21 1996 dx500/scripts/test2
-r--r--r--154/50    9212 Jul 16 22 21 1996 dx500 scripts/test3
-r--r--r--154/50   11189 Jul 30 20 42 1996 dx500 scripts/test4
-r--r--r--154/50    3678 Jul 16 22 21 1996 dx500 scripts/test5
-r--r--r--154/50   11832 Jul 16 22 21 1996 dx500 scripts/test6
-r--r--r--154/50    9971 Jul 18 00 40 1996 dx500 scripts/test7
-r--r--r--154/50    4168 Jul 18 00 40 1996 dx500/scripts/test8
-r--r--r--154/50    5801 Jul 16 22 21 1996 dx500 scripts/dsptest1
-rwxrwxr-x154/50       0 Jul 31 20 42 1996 dx500 api/
lrwxrwxrwx154/50       9 Aug 16 08 54 1996 dx500 api lib symbolic link to lib SUNOS
-rwxrwxr-x154/50       0 Jul 31 20 20 1996 dx500 api/include/
-rw-rw-r--154/50    7902 Jul 31 19 38 1996 dx500 api/include/Attr h
-rw-rw-r--154/50    9886 Jul 31 19 38 1996 dx500/api/include/Auth h
-rw-rw-r--154/50    9377 Jul 31 19 38 1996 dx500/api include/DapDsp h
-rw-rw-r--154/50   45626 Jul 31 19 38 1996 dx500 api include/Dapasn h
-rw-rw-r--154/50   11328 Jul 18 00 45 1996 dx500 api include/Dapidu h
-rw-rw-r--154/50   17948 Jul 31 19 38 1996 dx500/api/include/Dipidu h
-rw-rw-r--154/50   13587 Jul 31 19 38 1996 dx500/api include Dsp h
-rw-rw-r--154/50    3610 Jul 31 19 38 1996 dx500 api include/Info h
-rw-rw-r--154/50   17096 Jul 31 19 38 1996 dx500 api include MTS h
-rw-rw-r--154/50     636 Jul 31 19 38 1996 dx500 api include/RoseId h
-rw-rw-r--154/50    4413 Jul 31 19 38 1996 dx500 api include/Rupper h
-rw-rw-r--154/50     956 Jul 31 19 38 1996 dx500 api include SYNTAXES h
-rw-rw-r--154/50    6985 Jul 31 19 38 1996 dx500 api include Syx h
-r--r--r--154/50    6583 Jun 24 02 58 1996 dx500 api include/asnlsys h
-r--r--r--154/50    1409 Jul 16 22 20 1996 dx500 api include/ds h
-r--r--r--154/50    3101 Jun 24 02 58 1996 dx500/api include/queue h
-r--r--r--154/50     813 Jun 24 02 58 1996 dx500 api include/rstypes h
-rwxrwxr-x154/50       0 Aug  5 05 01 1996 dx500/api demo
-r--r--r--154/50     288 Jul 16 22 20 1996 dx500 api demo/Makefile PC
-r--r--r--154/50     276 Jul 16 22 20 1996 dx500 api demo/Makefile SCO
-r--r--r--154/50     286 Jul 16 22 20 1996 dx500 api demo/Makefile SOLARIS
-r--r--r--154/50     305 Jul 16 22 20 1996 dx500 api demo Makefile SUN
-r--r--r--154/50    9359 Jul 16 22 20 1996 dx500 api demo/attr c
-r--r--r--154/50   24921 Jul 16 22 20 1996 dx500 api demo conf c
-r--r--r--154/50    5675 Jul 16 22 20 1996 dx500 api demo demo c
-r--r--r--154/50    1860 Jul 16 22 20 1996 dx500 api demo demo h
-r--r--r--154/50   20799 Jul 16 22 20 1996 dx500 api demo req c
lrwxrwxrwx154/50      12 Aug 16 08 54 1996 dx500 api demo Makefile symbolic link to Makefile SUN
-rwxrwxr-x154/50       0 Aug  5 04 59 1996 dx500 api lib SUNOS
lrwxrwxrwx154/50       9 Aug 16 08 54 1996 dx500 api lib SUNOS lib symbolic link to lib SUNOS
-rwxrwxr-x154/50       0 Aug  5 04 59 1996 dx500 api lib SOLARIS
-rwxrwxr-x154/50       0 Jul 19 00 22 1996 dx500 utils
-rwxrwxr-x154/50       0 Aug 27 18 05 1996 dx500 utils cmip
-r-xr-xr-x154/50     644 Jul 16 22 21 1996 dx500 utils/cmip dxcmip
-r--r--r--154/50     508 Jul 18 23 44 1996 dx500 utils cmip README
-rwxrwxr-x154/50       0 Aug 27 18 04 1996 dx500 utils/dua/
-r--r--r--154/50     679 Jul 16 22 21 1996 dx500 utils dua init
-r--r--r--154/50     402 Feb 27 01 47 1995 dx500 utils dua bind
lrwxrwxrwx154/50      21 Aug 27 18 04 1996 dx500 utils dua dua symbolic link to       bin dua rfc1006
-rwxrwxr-x154/50       0 Aug 28 11 37 1996 dx500 utils/snmp
-r-xr-xr-x154/50     566 Jul 16 22 21 1996 dx500 utils/snmp dxsnmp
-r--r--r--154/50     910 Jul 16 22 21 1996 dx500 utils snmp README
lrwxrwxrwx154/50      38 Aug 28 11 37 1996 dx500 utils snmp dxmonitor symbolic link to         dxmonitor dxmonitor dxmonitor
-rwxrwxr-x154/50       0 Aug 27 18 04 1996 dx500 utils disp
-r--r--r--154/50     495 Jul 16 22 21 1996 dx500 utils disp README
```

```
-r--r--r--154/50      365 Jul 16 22:21 1996 dx500/utils/disp/dispCopy.cfg
lrwxrwxrwx154/50       18 Aug 27 18:04 1996 dx500/utils/disp/dispCopy symbolic link to ../../bin/dispCopy
-rwxrwxr-x154/50        0 Aug 27 18:06 1996 dx500/utils/actest/
-r--r--r--154/50     9434 Aug  5 04:12 1996 dx500/utils/actest/README
-r--r--r--154/50     5853 Aug  5 04:12 1996 dx500/utils/actest/actest.in
lrwxrwxrwx154/50       16 Aug 27 18:06 1996 dx500/utils/actest/actest symbolic link to ../../bin/actest
-r--r--r--154/50     2534 Feb 26 01:48 1996 dx500/BETA.AGREEMENT
-rwxrwxr-x154/50        0 Aug 16 08:54 1996 rstack/
-rwxrwxr-x154/50        0 Aug 27 17:31 1996 rstack/asn/
-rw-rw-r--154/50    14247 Aug 27 17:31 1996 rstack/asn/STACK.asn
-rw-rw-r--154/50    42956 Aug 27 17:31 1996 rstack/asn/DAP.asn
-rw-rw-r--154/50    14420 Aug 27 17:31 1996 rstack/asn/CMIP.asn
-rw-rw-r--154/50    10853 Aug 27 17:31 1996 rstack/asn/LDAP.asn
-r--r--r--154/50     3703 Jun 24 02:57 1996 rstack/asn/acse.asn
-r--r--r--154/50     4213 Jun 24 02:57 1996 rstack/asn/basicAC.asn
-r--r--r--154/50    13639 Jun 24 02:57 1996 rstack/asn/cmip.asn
-r--r--r--154/50    18946 Jun 24 02:57 1996 rstack/asn/dap.asn
-r--r--r--154/50     2693 Jun 24 02:57 1996 rstack/asn/dapdsp.asn
-r--r--r--154/50      781 Jun 24 02:57 1996 rstack/asn/defs.asn
-r--r--r--154/50     7579 Jun 24 02:57 1996 rstack/asn/disp.asn
-r--r--r--154/50     3147 Jun 24 02:57 1996 rstack/asn/dop.asn
-r--r--r--154/50     4902 Jun 24 02:57 1996 rstack/asn/dsp.asn
-r--r--r--154/50     1784 Jun 24 02:57 1996 rstack/asn/info.asn
-r--r--r--154/50    10072 Jun 24 02:57 1996 rstack/asn/ldap.asn
-rw-rw-r--154/50     1754 Aug 21 12:49 1996 rstack/asn/makefile
-r--r--r--154/50     5057 Jun 24 02:57 1996 rstack/asn/pres.asn
-r--r--r--154/50     2189 Jun 24 02:57 1996 rstack/asn/rose.asn
-r--r--r--154/50      607 Jun 24 02:57 1996 rstack/asn/roseId.asn
-r--r--r--154/50     2517 Jun 24 02:57 1996 rstack/asn/upper.asn
-rwxrwxr-x154/50        0 Aug 27 17:36 1996 rstack/cmip/
-r--r--r--154/50    15317 Jun 24 02:57 1996 rstack/cmip/cm_agent.c
-r--r--r--154/50     8451 Jun 24 02:57 1996 rstack/cmip/cmip.c
-r--r--r--154/50     8581 Jun 24 02:57 1996 rstack/cmip/cmip_mib.c
-r--r--r--154/50     2124 Jun 24 02:57 1996 rstack/cmip/cmip_mib.h
-r--r--r--154/50     1703 Jun 24 02:57 1996 rstack/cmip/cmipi.c
-r--r--r--154/50     2170 Jun 24 02:57 1996 rstack/cmip/cmipr.c
-r--r--r--154/50    17787 Jun 24 02:57 1996 rstack/cmip/dsa_mib.c
-r--r--r--154/50      488 Jun 24 02:57 1996 rstack/cmip/makefile
-rw-rw-r--154/50    95895 Aug 27 17:31 1996 rstack/cmip/CMIP_i.c
-rw-rw-r--154/50    87352 Aug 27 17:31 1996 rstack/cmip/CMIP_r.c
-rw-rw-r--154/50    36412 Aug 27 17:35 1996 rstack/cmip/cm_agent.o
-rw-rw-r--154/50    35452 Aug 27 17:35 1996 rstack/cmip/cmip_mib.o
-rw-rw-r--154/50    34756 Aug 27 17:35 1996 rstack/cmip/cmip.o
-rw-rw-r--154/50    27200 Aug 27 17:35 1996 rstack/cmip/cmipr.o
-rw-rw-r--154/50    94724 Aug 27 17:36 1996 rstack/cmip/CMIP_r.o
-rw-rw-r--154/50    50332 Aug 27 17:36 1996 rstack/cmip/dsa_mib.o
-rw-rw-r--154/50    25560 Aug 27 17:36 1996 rstack/cmip/cmipi.o
-rw-rw-r--154/50   102380 Aug 27 17:36 1996 rstack/cmip/CMIP_i.o
-rwxrwxr-x154/50        0 Aug 27 17:34 1996 rstack/x500-fp/
-r--r--r--154/50      379 Jun 24 02:57 1996 rstack/x500-fp/makefile
-rw-rw-r--154/50    39324 Aug 27 17:31 1996 rstack/x500-fp/DapDsp_i.c
-rw-rw-r--154/50   217337 Aug 27 17:31 1996 rstack/x500-fp/Dapasn_i.c
-rw-rw-r--154/50    68551 Aug 27 17:31 1996 rstack/x500-fp/Dispasn_i.c
-rw-rw-r--154/50    32413 Aug 27 17:31 1996 rstack/x500-fp/Dopasn_i.c
-rw-rw-r--154/50    73101 Aug 27 17:31 1996 rstack/x500-fp/Dsp_i.c
-rw-rw-r--154/50    10895 Aug 27 17:31 1996 rstack/x500-fp/Info_i.c
```

```
-rw-rw-r--154/50      916 Aug 27 17:31 1996 rstack/x500-fp/RoseId_i.c
-rw-rw-r--154/50    48200 Aug 27 17:33 1996 rstack/x500-fp/DapDsp_i.o
-rw-rw-r--154/50   246116 Aug 27 17:34 1996 rstack/x500-fp/Dapasn_i.o
-rw-rw-r--154/50   102508 Aug 27 17:34 1996 rstack/x500-fp/Dsp_i.o
-rw-rw-r--154/50    15156 Aug 27 17:34 1996 rstack/x500-fp/Info_i.o
-rw-rw-r--154/50     3812 Aug 27 17:34 1996 rstack/x500-fp/RoseId_i.o
-rw-rw-r--154/50    74600 Aug 27 17:34 1996 rstack/x500-fp/Dispasn_i.o
-rw-rw-r--154/50    44512 Aug 27 17:34 1996 rstack/x500-fp/Dopasn_i.o
-rwxrwxr-x154/50        0 Aug 27 17:37 1996 rstack/ac/
-rw-rw-r--154/50    57025 Aug 16 09:11 1996 rstack/ac/ac.c
-r--r--r--154/50     8839 Aug  5 04:12 1996 rstack/ac/ac.h
-r--r--r--154/50     8575 Jul 30 20:46 1996 rstack/ac/acdump.c
-r--r--r--154/50    32908 Aug  5 04:12 1996 rstack/ac/actest.c
-r--r--r--154/50     5853 Aug  5 04:12 1996 rstack/ac/actest.in
-r--r--r--154/50     9434 Aug  5 04:12 1996 rstack/ac/actest.readme
-r--r--r--154/50     6132 Jul 19 01:04 1996 rstack/ac/basicAC.asn
-r--r--r--154/50      456 Aug  5 04:12 1996 rstack/ac/makefile
-rw-rw-r--154/50      968 Aug 27 17:36 1996 rstack/ac/Dapasn.h
-rw-rw-r--154/50    66600 Aug 27 17:36 1996 rstack/ac/ac.o
-r--r--r--154/50    29125 Aug  5 04:12 1996 rstack/ac/acadd.c
-rw-rw-r--154/50    14198 Aug 27 17:36 1996 rstack/ac/BasicAC.h
-rw-rw-r--154/50    47684 Aug 27 17:36 1996 rstack/ac/acadd.o
-rw-rw-r--154/50    26120 Aug 27 17:36 1996 rstack/ac/acdump.o
-rw-rw-r--154/50    51084 Aug 27 17:37 1996 rstack/ac/actest.o
-rwxrwxr-x154/50   327680 Aug 27 17:37 1996 rstack/ac/actest
-rwxrwxr-x154/50        0 Aug 27 17:31 1996 rstack/include/
-rw-rw-r--154/50     6648 Aug 24 19:43 1996 rstack/include/asnlsys.h
-r--r--r--154/50     1286 Jun 24 02:58 1996 rstack/include/bufflib.h
-r--r--r--154/50     5339 Jun 24 02:58 1996 rstack/include/cm.h
-r--r--r--154/50     4004 Jun 24 02:58 1996 rstack/include/cmip.h
-r--r--r--154/50      861 Jun 24 02:58 1996 rstack/include/commslib.h
-r--r--r--154/50      811 Jun 24 02:58 1996 rstack/include/config.h
-r--r--r--154/50     1528 Jun 24 02:58 1996 rstack/include/dapi.h
-r--r--r--154/50     1341 Jun 24 02:58 1996 rstack/include/network.h
-r--r--r--154/50      647 Jun 24 02:58 1996 rstack/include/objidv.h
-r--r--r--154/50     3101 Jun 24 02:58 1996 rstack/include/queue.h
-r--r--r--154/50      828 Jun 24 02:58 1996 rstack/include/rfc1277.h
-r--r--r--154/50     1886 Jun 24 02:58 1995 rstack/include/rsinfo.h
-r--r--r--154/50     4800 Jun 24 02:58 1996 rstack/include/rstack.h
-r--r--r--154/50      813 Jun 24 02:58 1996 rstack/include/rstypes.h
-r--r--r--154/50     1419 Jun 24 02:58 1996 rstack/include/snmp.h
-r--r--r--154/50      666 Jun 24 02:58 1996 rstack/include/timer.h
-r--r--r--154/50     3481 Jul 30 20:47 1996 rstack/include/x500.h
-r--r--r--154/50     2239 Jun 24 02:58 1996 rstack/include/xm.h
-rw-rw-r--154/50     9377 Aug 27 17:31 1996 rstack/include/DapDsp.h
-rw-rw-r--154/50    45626 Aug 27 17:31 1996 rstack/include/Dapasn.h
-rw-rw-r--154/50    22969 Aug 27 17:31 1996 rstack/include/Dispasn.h
-rw-rw-r--154/50     7655 Aug 27 17:31 1996 rstack/include/Dopasn.h
-rw-rw-r--154/50    12587 Aug 27 17:31 1996 rstack/include/Dsp.h
-rw-rw-r--154/50     3610 Aug 27 17:31 1996 rstack/include/Info.h
-rw-rw-r--154/50      636 Aug 27 17:31 1996 rstack/include/RoseId.h
-rw-rw-r--154/50     4413 Aug 27 17:31 1996 rstack/include/Rupper.h
-rw-rw-r--154/50    43922 Aug 27 17:31 1996 rstack/include/CMIP.h
-rwxrwxr-x154/50        0 Aug 27 17:33 1996 rstack/network/
-r--r--r--154/50    25861 Jun 24 02:58 1996 rstack/network/Hsun.c
-r--r--r--154/50    16110 Jul 30 20:47 1996 rstack/network/Htcp.c
```

```
Sep 26 15:56 1997  listing 3 Page 23

-r--r--r--154/50     257 Jun 24 02 58 1996 rstack/network/makefile
-rw-rw-r--154/50   31804 Aug 27 17 33 1996 rstack/network/Http.o
-rwxrwxr-x154/50       0 Aug 27 17 35 1996 rstack/snmp
-r--r--r--154/50    3666 Jun 24 02 58 1996 rstack/snmp/m_dsa.c
-r--r--r--154/50    2799 Jun 24 02 58 1996 rstack/snmp/m_system.c
-r--r--r--154/50     450 Jun 24 02 58 1996 rstack/snmp/makefile
-r--r--r--154/50   14102 Jun 24 02 58 1996 rstack/snmp/mibsupp.c
-r--r--r--154/50    2609 Jun 24 02 58 1996 rstack/snmp/mibsupp.h
-r--r--r--154/50   11659 Jun 24 02 58 1996 rstack/snmp/snmp_agent.c
-r--r--r--154/50   24247 Jun 24 02 58 1996 rstack/snmp/snmplib.c
-r--r--r--154/50    4489 Jun 24 02 58 1996 rstack/snmp/snmplib.h
-r--r--r--154/50    1161 Jun 24 02 58 1996 rstack/snmp/system.asn1
-r--r--r--154/50    2826 Jul 30 20 47 1996 rstack/snmp/udplib.c
-r--r--r--154/50   18715 Jun 24 02 58 1996 rstack/snmp/x500.asn1
-rw-rw-r--154/50    1105 Jul 18 00 54 1996 rstack/snmp/mib.h
-rw-rw-r--154/50    5049 Jul 18 00 54 1996 rstack/snmp/mib.c
-rw-rw-r--154/50   41468 Aug 27 17 35 1996 rstack/snmp/snmplib.o
-rw-rw-r--154/50   28204 Aug 27 17 35 1996 rstack/snmp/mibsupp.o
-rw-rw-r--154/50   10592 Aug 27 17 35 1996 rstack/snmp/udplib.o
-rw-rw-r--154/50   22848 Aug 27 17 35 1996 rstack/snmp/snmp_agent.o
-rw-rw-r--154/50    9236 Aug 27 17 35 1996 rstack/snmp/m_system.o
-rw-rw-r--154/50   10932 Aug 27 17 35 1996 rstack/snmp/m_dsa.o
-rw-rw-r--154/50   13540 Aug 27 17 35 1996 rstack/snmp/mib.o
-rwxrwxr-x154/50       0 Aug 27 17 33 1996 rstack/stack/
-r--r--r--154/50     500 Jun 24 02 58 1996 rstack/stack/makefile
-r--r--r--154/50   21798 Jul 30 20 47 1996 rstack/stack/rdebug.c
-r--r--r--154/50   19671 Jun 24 02 58 1996 rstack/stack/rpres.c
-r--r--r--154/50   18880 Jun 24 02 58 1996 rstack/stack/rsess.c
-r--r--r--154/50   39556 Jun 24 02 58 1996 rstack/stack/rstack.c
-r--r--r--154/50    7047 Jun 24 02 58 1996 rstack/stack/rstacki.h
-r--r--r--154/50   17335 Jun 24 02 58 1996 rstack/stack/rtran.c
-rw-rw-r--154/50    7331 Aug 27 17 31 1996 rstack/stack/Racse.h
-rw-rw-r--154/50   14976 Aug 27 17 31 1996 rstack/stack/Rpres.h
-rw-rw-r--154/50    4873 Aug 27 17 31 1996 rstack/stack/Rrose.h
-rw-rw-r--154/50   23244 Aug 27 17 31 1996 rstack/stack/Racse_i.c
-rw-rw-r--154/50   36890 Aug 27 17 31 1996 rstack/stack/Rpres_i.c
-rw-rw-r--154/50   13354 Aug 27 17 31 1996 rstack/stack/Rrose_i.c
-rw-rw-r--154/50   58100 Aug 27 17 32 1996 rstack/stack/rstack.o
-rw-rw-r--154/50   43564 Aug 27 17 32 1996 rstack/stack/rpres.o
-rw-rw-r--154/50   37036 Aug 27 17 32 1996 rstack/stack/rsess.o
-rw-rw-r--154/50   30548 Aug 27 17 32 1996 rstack/stack/rtran.o
-rw-rw-r--154/50   45704 Aug 27 17 32 1996 rstack/stack/rdebug.o
-rw-rw-r--154/50   28664 Aug 27 17 32 1996 rstack/stack/Racse_i.o
-rw-rw-r--154/50   14784 Aug 27 17 32 1996 rstack/stack/Rrose_i.o
-rw-rw-r--154/50   38964 Aug 27 17 33 1996 rstack/stack/Rpres_i.o
-rwxrwxr-x154/50       0 Aug 27 17 32 1996 rstack/support
-rw-rw-r--154/50   43273 Jun 24 02 58 1996 rstack/support/asnprocs.c
-r--r--r--154/50    1775 Jun 24 02 58 1996 rstack/support/bufflib.c
-r--r--r--154/50   11476 Jun 24 02 58 1996 rstack/support/commslib.c
-r--r--r--154/50    5756 Jun 24 02 58 1996 rstack/support/config.c
-r--r--r--154/50     427 Jun 24 02 58 1996 rstack/support/makefile
-r--r--r--154/50    1375 Jun 24 02 58 1996 rstack/support/objidv.c
-r--r--r--154/50    2110 Jun 24 02 58 1996 rstack/support/queue.c
-r--r--r--154/50    5910 Jun 24 02 58 1996 rstack/support/rfc1277.c
-r--r--r--154/50    4947 Jun 24 02 58 1996 rstack/support/rsinfo.c
-r--r--r--154/50    4290 Jun 24 02 58 1996 rstack/support/timer.c
```

```
Sep 26 15:56 1997  listing 3 Page 24

-r--r--r--154/50   13253 Jun 24 02 58 1996 rstack/support/xm.c
-r--r--r--154/50     762 Jun 24 02 58 1996 rstack/support/xmalloc.c
-rw-rw-r--154/50   74448 Aug 27 17 32 1996 rstack/support/asnprocs.o
-rw-rw-r--154/50   11248 Aug 27 17 32 1996 rstack/support/rsinfo.o
-rw-rw-r--154/50    4336 Aug 27 17 32 1996 rstack/support/bufflib.o
-rw-rw-r--154/50    6264 Aug 27 17 32 1996 rstack/support/timer.o
-rw-rw-r--154/50    3056 Aug 27 17 32 1996 rstack/support/queue.o
-rw-rw-r--154/50   16812 Aug 27 17 32 1996 rstack/support/commslib.o
-rw-rw-r--154/50   15712 Aug 27 17 32 1996 rstack/support/xm.o
-rw-rw-r--154/50    2580 Aug 27 17 32 1996 rstack/support/xmalloc.o
-rw-rw-r--154/50    7080 Aug 27 17 32 1996 rstack/support/objidv.o
-rw-rw-r--154/50   10632 Aug 27 17 32 1996 rstack/support/rfc1277.o
-rw-rw-r--154/50   12516 Aug 27 17 32 1996 rstack/support/config.o
-rwxrwxr-x154/50       0 Aug 27 17 35 1996 rstack/ldap
-r--r--r--154/50   88002 Jun 24 02 58 1996 rstack/ldap/ldap.c
-r--r--r--154/50     348 Jun 24 02 58 1996 rstack/ldap/makefile
-rw-rw-r--154/50   15500 Aug 27 17 31 1996 rstack/ldap/LDAP.h
-rw-rw-r--154/50   48480 Aug 27 17 31 1996 rstack/ldap/LDAP_i.c
-rw-rw-r--154/50  156628 Aug 27 17 35 1996 rstack/ldap/ldap.o
-rw-rw-r--154/50   50236 Aug 27 17 35 1996 rstack/ldap/LDAP_i.o
-rwxrwxr-x154/50       0 Aug 27 17 33 1996 rstack/x500
-r--r--r--154/50    9622 Jul 30 20 47 1996 rstack/x500/dapi.c
-r--r--r--154/50    7212 Jul 30 20 47 1996 rstack/x500/dapr.c
-r--r--r--154/50    8195 Jun 24 02 58 1996 rstack/x500/disp.c
-r--r--r--154/50    6342 Jun 24 02 58 1996 rstack/x500/dop.c
-r--r--r--154/50    9637 Jul 30 20 47 1996 rstack/x500/dsp.c
-r--r--r--154/50     318 Jun 24 02 58 1996 rstack/x500/makefile
-r--r--r--154/50    6910 Jun 24 02 58 1996 rstack/x500/x500.c
-rw-rw-r--154/50   38560 Aug 27 17 33 1996 rstack/x500/x500.o
-rw-rw-r--154/50   46184 Aug 27 17 33 1996 rstack/x500/dapr.o
-rw-rw-r--154/50   49884 Aug 27 17 33 1996 rstack/x500/dsp.o
-rw-rw-r--154/50   32772 Aug 27 17 33 1996 rstack/x500/disp.o
-rw-rw-r--154/50   23920 Aug 27 17 33 1996 rstack/x500/dop.o
-rw-rw-r--154/50   41100 Aug 27 17 33 1996 rstack/x500/dapi.o
-r--r--r--154/50     572 Jun 24 02 58 1996 rstack/makefile
-rwxrwxr-x154/50       0 Aug 27 17 36 1996 rstack/lib
-rw-rw-r--154/50  168212 Aug 27 17 32 1996 rstack/lib/support.a
-rw-rw-r--154/50  300442 Aug 27 17 33 1996 rstack/lib/rstack.a
-rw-rw-r--154/50   31804 Aug 27 17 33 1996 rstack/lib/Http.o
-rw-rw-r--154/50  233310 Aug 27 17 33 1996 rstack/lib/x500.a
-rw-rw-r--154/50  343128 Aug 27 17 34 1996 rstack/lib/x500-fp.a
-rw-rw-r--154/50  209278 Aug 27 17 35 1996 rstack/lib/ldap.a
-rw-rw-r--154/50  138452 Aug 27 17 35 1996 rstack/lib/snmp.a
-rw-rw-r--154/50  282192 Aug 27 17 36 1996 rstack/lib/cmip.a
-rw-rw-r--154/50  165356 Aug 27 17 36 1996 rstack/lib/cmipi.a
```

MISSION OF AFRL/INFORMATION DIRECTORATE (IF)